John L. Cooper (State Bar No. 050324)
jcooper@fbm.com
Jeffrey M. Fisher (State Bar No. 155284)
jfisher@fbm.com
Helen E. Dutton (State Bar No. 235558)
hdutton@fbm.com
Farella Braun & Martel LLP
235 Montgomery Street, 17th Floor
San Francisco, CA 94104
Telephone: (415) 954-4400
Facsimile: (415) 954-4480

Attorneys for Defendants
TECHNOLOGY PROPERTIES LIMITED
and ALLIACENSE LIMITED

Charles T. Hoge, Esq. (State Bar No. 110696)
choge@knlh.com
Kirby Noonan Lance & Hoge
35 Tenth Avenue
San Diego, CA 92101
Telephone: (619) 231-8666
Facsimile: (619) 231-9593

Attorneys for Defendant
PATRIOT SCIENTIFIC CORPORATION

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN JOSE DIVISION

ACER, INC., ACER AMERICA CORPORATION and GATEWAY, INC.,

Plaintiff,

v.

TECHNOLOGY PROPERTIES LIMITED, PATRIOT SCIENTIFIC CORPORATION, and ALLIACENSE LIMITED,

Defendants.

Case No. 5:08-cv-00877 JF

DECLARATION OF JEFFREY M. FISHER IN SUPPORT OF DEFENDANTS' REPLY IN SUPPORT OF MOTION TO DISMISS OR, IN THE ALTERNATIVE, TO TRANSFER VENUE; MEMORANDUM OF POINTS AND AUTHORITIES IN SUPPORT THEREOF

Date: August 1, 2008
Time: 9:00 a.m.
Dept: Courtroom 3, 5th Floor
Before: Honorable Jeremy Fogel

I, Jeffrey M. Fisher, declare the following:

1. I am an attorney licensed to practice law in the State of California and am a partner with the firm of Farella Braun & Martel LLP, counsel for Defendants Technology Properties Limited ("TPL") and Alliacense Limited in this action. I have personal knowledge of the facts set forth below and, if called upon to do so, could and would testify competently thereto.

2. On April 25, 2008, TPL filed complaints in the Eastern District of Texas alleging that Acer, Inc., Acer America Corporation and Gateway, Inc. (collectively "Acer") infringe certain MMP patents. These patents include U.S. Patent Nos. 5,809,336 ("the '336 patent"), 5,784,584 ("the '584 patent"), 5,440,749 ("the '749 patent"), and 6,598,148 ("the '148 patent"). Attached hereto as Exhibits A and B are true and correct copies of the complaints in these actions, both of which have been assigned to Judge Ward.

3. On June 4, 2008, TPL filed an additional complaint in the Eastern District of Texas alleging that Acer infringes another MMP patent, U.S. Patent No. 5,530,890 ("the '890 patent"). Attached hereto as Exhibit C is a true and correct copy of the complaint in this action, which has also been assigned to Judge Ward.

4. On July 16, 2008, John Cooper and I participated in a meet and confer conference with Acer's counsel (Mr. Davis) concerning the Joint Case Management Conference Statement to be filed in connection with the Case Management Conference scheduled for August 1, 2008. During the course of this meet and confer conference, Mr. Davis stated that Acer intends to amend its complaint to add the '148 and '890 patents in this action.

5. Attached hereto as Exhibit D is a true and correct copy of the docket for *Mosaid Techs. Inc. v. Hynix Semiconductor Inc.*, Docket # 6:05-cv-00013-LED (E.D. Tex.)(Davis, J.)

6. Attached hereto as Exhibit E is a true and correct copy of the docket for *Mosaid Techs. Inc. v. Infineon Techs N. Am. Corp.*, Docket # 6:05-cv-00120-LED (E.D.Tex.) (Davis, J.)

7. Attached hereto as Exhibit F is a true and correct copy of the docket for *Guardian Media Techs. Ltd. v. LG Elecs. Inc.*, Docket # 2:07-cv-00692-R-RC (C.D.Cal.) (Manuel, J.)

8. Attached hereto as Exhibit G is a true and correct copy of a chart showing the MMP patents that are pending in the actions in the Northern District of California and Eastern

District of Texas and the MMP patents that have already been litigated and construed by Judge Ward in the Eastern District of Texas.

9. Attached hereto as Exhibit H is a true and correct copy of the docket for *Acer Am. Corp. v. Hon Hai Precision Indus. Co*., Docket # 2:07-cv-00181-TJW-CE (E.D.Tex.) (Ward, J.)

10. Attached hereto as Exhibit I is a true and correct copy of the docket for *Acer, Inc. v. Hewlett-Packard Co.*, Docket # 3:07-cv-00620-bbc (W.D. Wis.) (Crabb, J.)

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct. Executed this 18th day of July, 2008, at San Francisco, California.

/s/ Jeffrey M. Fisher
Jeffrey M. Fisher

EXHIBIT A

TO THE DECLARATION OF JEFFREY M. FISHER ISO DEFENDANTS' REPLY ISO MOTION TO DISMISS OR, IN THE ALTERNATIVE, TO TRANSFER VENUE

IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

(1) TECHNOLOGY PROPERTIES LIMITED and (2) PATRIOT SCIENTIFIC CORPORATION,

Plaintiffs,

vs.

(1) ACER, INC.
(2) ACER AMERICA CORPORATION
(3) GATEWAY, INC.

Defendants.

CASE NO. 2-08CV-176 TJW

Jury Trial Demanded

COMPLAINT FOR PATENT INFRINGEMENT AND DEMAND FOR JURY TRIAL

Plaintiffs, Technology Properties Limited ("TPL") and Patriot Scientific Corporation ("Patriot"), (collectively "Plaintiffs"), allege the following in support of their Complaint for Patent Infringement and Demand for Jury Trial ("Complaint") against Defendants, Acer, Inc, ("Acer"), Acer America Corporation ("Acer America") and Gateway, Inc. ("Gateway").

PARTIES

1. Plaintiff, Technology Properties Limited ("TPL") is a corporation duly organized and existing under the laws of the State of California and maintains its principal place of business in Cupertino, California.

2. Plaintiff, Patriot Scientific Corporation ("Patriot") is a corporation duly organized and existing under the laws of the State of Delaware and maintains its principal place of business in Carlsbad, California.

3. Upon information and belief, Defendant Acer, Inc. is a Taiwan corporation with its principal place of business in Taipei, Taiwan, R.O.C

4 Upon information and belief, Defendant Acer America Corporation is a California corporation with its principal place of business in San Jose, California

5. Upon information and belief, Defendant Gateway, Inc. is a Delaware corporation with its principal place of business in Irvine, California. Gateway is a wholly-owned subsidiary of Acer.

JURISDICTION

6 This Court has subject matter jurisdiction over this action pursuant to 28 U.S.C §§ 1331, 1338(a) because this action arises under the patent laws of the United States, including 35 U.S.C. §§ 101, *et seq* and 271, *et seq*. This Court has personal jurisdiction over Defendants because they each infringe Plaintiffs' patents by offering on their websites infringing products to their users and/or customers who reside in, or may be found in, the Eastern District of Texas Further, each Defendant has actually transacted business with users of their websites in the Eastern District of Texas.

VENUE

7. Venue is proper in this judicial district under 28 U.S.C. §§ 1391(b) and (c) and 1400(b) because Defendants have each committed acts of infringement in this district.

GENERAL ALLEGATIONS

8. On September 15, 1998, United States Patent No. 5,809,336 ('336 Patent") entitled "High Performance Microprocessor Having Variable Speed System Clock" was duly and legally issued All rights and interest in the '336 Patent are co-owned by TPL and Patriot Scientific Corporation. TPL has the sole and exclusive right and obligation to license and

enforce the '336 Patent A true and correct copy of the '336 Patent is attached hereto as Exhibit A.

9. On August 8, 1995, United States Patent No 5,440,749 ('749 Patent") entitled "High Performance, Low Cost Microprocessor Architecture" was duly and legally issued All rights and interest in the '749 Patent are co-owned by TPL and Patriot . TPL has the sole and exclusive right and obligation to license and enforce the '749 Patent. A true and correct copy of the '749 Patent is attached hereto as Exhibit B

10 On July 22, 2003, United States Patent No. 6,598,148 ('148 Patent") entitled "High Performance Microprocessor Having Variable Speed System Clock" was duly and legally issued. All rights and interest in the '148 Patent are co-owned by TPL and Patriot TPL has the sole and exclusive right and obligation to license and enforce the '148 Patent. A true and correct copy of the '148 Patent is attached hereto as Exhibit C

COUNT 1

(Patent Infringement Against Acer, Inc)

11 Paragraphs 1-10 of the Complaint set forth above are incorporated herein by reference.

12. Upon information and belief Defendant Acer has infringed and continues to infringe under 35 U S C. § 271 the '336 Patent, the '749 Patent, and the '148 Patent (collectively "the patents-in-suit).

13. Acer's acts of infringement have caused damage to Plaintiffs. Under 35 U.S.C. § 284, Plaintiffs are entitled to recover from Acer the damages sustained by Plaintiffs as a result of its infringement of the patents-in-suit. Acer's infringement of Plaintiffs' exclusive rights under the patents-in-suit will continue to damage Plaintiffs' business, causing irreparable harm,

for which there is no adequate remedy at law, unless enjoined by this Court under 35 U.S.C. § 283.

14 Plaintiffs allege, on information and belief, that Acer's acts of infringement were willful and deliberate.

COUNT 2

(Patent Infringement Against Acer America Corporation)

15 Paragraphs 1-10 of the Complaint set forth above are incorporated herein by reference

16 Upon information and belief Defendant Acer America has infringed and continues to infringe under 35 U S C § 271 the '336 Patent, the '749 Patent, and '148 Patent (collectively "the patents-in-suit).

17. Acer America's acts of infringement have caused damage to Plaintiffs. Under 35 U.S.C. § 284, Plaintiffs are entitled to recover from Acer America the damages sustained by Plaintiffs as a result of its infringement of the patents-in-suit Acer America's infringement of Plaintiffs' exclusive rights under the patents-in-suit will continue to damage Plaintiffs' business, causing irreparable harm, for which there is no adequate remedy at law, unless enjoined by this Court under 35 U S.C. § 283.

18 Plaintiffs allege, on information and belief, that Acer America's acts of infringement were willful and deliberate.

COUNT 3

(Patent Infringement Against Gateway, Inc.)

19. Paragraphs 1-10 of the Complaint set forth above are incorporated herein by reference.

20. Upon information and belief Defendant Gateway has infringed and continues to infringe under 35 U.S.C. § 271 the '336 Patent, the '749 Patent, and the '148 Patent (collectively, "the patents-in-suit).

21. Gateway's acts of infringement have caused damage to Plaintiffs. Under 35 U.S.C. § 284, Plaintiffs are entitled to recover from Gateway the damages sustained by Plaintiffs as a result of its infringement of the patents-in-suit. Gateway's infringement of Plaintiffs' exclusive rights under the patents-in-suit will continue to damage Plaintiffs' business, causing irreparable harm, for which there is no adequate remedy at law, unless enjoined by this Court under 35 U.S.C. § 283.

22. Plaintiffs allege, on information and belief, that Gateway's acts of infringement were willful and deliberate.

PRAYER FOR RELIEF

WHEREFORE, Plaintiffs respectfully request that this Court enter judgment against Defendants as follows:

A. For judgment that Defendants Acer, Inc., Acer America Corporation, and Gateway, Inc. have infringed and continue to infringe the patents-in-suit;

B. For permanent injunctions under 35 U.S.C. § 283 against Defendants and their directors, officers, employees, agents, subsidiaries, parents, attorneys, and all persons acting in concert, on behalf of, in joint venture, or in partnership with Defendants from further acts of infringement;

C. For damages to be paid by Defendants adequate to compensate Plaintiffs for their infringement, including interests, costs and disbursements as the Court may deem appropriate under 35 U.S.C. § 284;

D. For judgment finding that Defendants' infringement was willful and deliberate, entitling Plaintiffs to increased damages under 35 U.S.C. § 284;

E. For judgment finding this to be an exceptional case against Defendants and awarding Plaintiffs attorney fees under 35 U.S.C. § 285; and,

F. For such other and further relief at law and in equity as the court may deem just and proper.

DEMAND FOR JURY TRIAL

Pursuant to the Federal Rules of Civil Procedure Rule 38, Plaintiffs hereby demand a jury trial on all issues triable by jury.

Dated: April 25, 2008

Respectfully submitted,

By: [signature]

S. Calvin Capshaw
State Bar No. 03783900
Email: ccapshaw@capshawlaw.com
Elizabeth L. DeRieux
State Bar No. 05770585
Email: ederieux@capshawlaw.com
Capshaw DeRieux, LLP
1127 Judson Road, Suite 220
Longview, TX 75601
Telephone: (903) 236-9800
Facsimile: (903) 236-8787

John L. Cooper
California Bar No. 50324
Email: jcooper@fbm.com
Jeffrey M. Fisher
California Bar No. 155284
Email: jfisher@fbm.com
Helen Dutton
California Bar No. 235558
hdutton@fbm.com

Farella Braun + Martel, LLP
Russ Building
235 Montgomery Street
San Francisco, CA 94104
Telephone: (415) 954-4400
Facsimile: (415) 954-4480

ATTORNEYS FOR PLAINTIFF
TECHNOLOGY PROPERTIES LIMITED

By: Charles T Hoge by permission EJD

Robert M. Parker
State Bar No. 15498000
Email: rmparker@pbatyler.com
Robert Christopher Bunt
State Bar No 00787165
Email: rcbunt@pbatyler.com
Parker, Bunt & Ainsworth, P.C.
100 East Ferguson, Ste. 1114
Tyler, TX 75702
Telephone: (903) 531-3535
Facsimile: (903) 533-9687

Charles T Hoge
California Bar No 110696
Email: choge@knlh.com
Kirby Noonan Lance & Hoge, LLP
350 Tenth Avenue, Suite 1300
San Diego, CA 92101
Telephone: (619) 231-8666
Facsimile: (619) 231-9593

ATTORNEYS FOR PLAINTIFF
PATRIOT SCIENTIFIC CORPORATION

US005809336A

United States Patent [19]

Moore et al.

[11] **Patent Number: 5,809,336**

[45] **Date of Patent: Sep. 15, 1998**

[54] **HIGH PERFORMANCE MICROPROCESSOR HAVING VARIABLE SPEED SYSTEM CLOCK**

[75] Inventors: **Charles H. Moore**, Woodside; **Russell H. Fish, III**, Mt. View, both of Calif.

[73] Assignee: **Patriot Scientific Corporation**, San Diego, Calif.

[21] Appl. No.: **484,918**

[22] Filed: **Jun. 7, 1995**

Related U.S. Application Data

[62] Division of Ser. No. 389,334, Aug. 3, 1989, Pat. No. 5,440,749.

[51] **Int. Cl.**^{6} **G06F 1/04**

[52] **U.S. Cl.** **395/845**

[58] **Field of Search** 395/500, 551, 395/555, 845

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,967,104	6/1976	Brantingham	364/709.09
3,980,993	9/1976	Bredart et al.	395/550
4,003,028	1/1977	Bennett et al.	395/742
4,042,972	8/1977	Gruner et al.	395/389
4,050,096	9/1977	Bennett	395/494
4,112,490	9/1978	Pohlman et al.	395/287
4,315,308	2/1982	Jackson	395/853
4,338,675	7/1982	Palmer	364/748
4,398,265	8/1983	Puhl et al.	395/882
4,453,229	6/1984	Schaire	395/250
4,503,500	3/1985	Magan	395/800
4,539,655	9/1985	Trussell et al.	395/280
4,553,201	11/1985	Pollack	395/183.22
4,627,082	12/1986	Pelgrom et al.	377/63
4,670,837	6/1987	Sheets	395/550
4,680,698	7/1987	Edwards et al.	395/800
4,761,763	8/1988	Hicks	395/286
5,414,862	5/1995	Suzuki et al.	395/750

Primary Examiner—David Y. Eng
Attorney, Agent, or Firm—Cooley Godward LLP

[57] **ABSTRACT**

A high performance, low cost microprocessor system having a variable speed system clock is disclosed herein. The microprocessor system includes an integrated circuit having a central processing unit and a ring oscillator variable speed system clock for clocking the microprocessor. The central processing unit and ring oscillator variable speed system clock each include a plurality of electronic devices of like type, which allows the central processing unit to operate at a variable processing frequency dependent upon a variable speed of the ring oscillator variable speed system clock. The microprocessor system may also include an input/output interface connected to exchange coupling control signals, address and data with the central processing unit. The input/output interface is independently clocked by a second clock connected thereto.

10 Claims, 19 Drawing Sheets

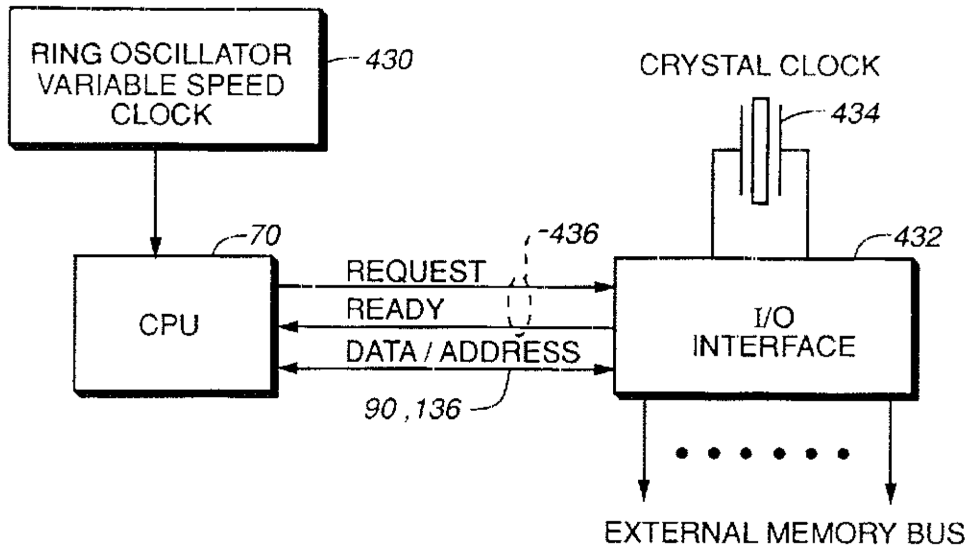


EXHIBIT A

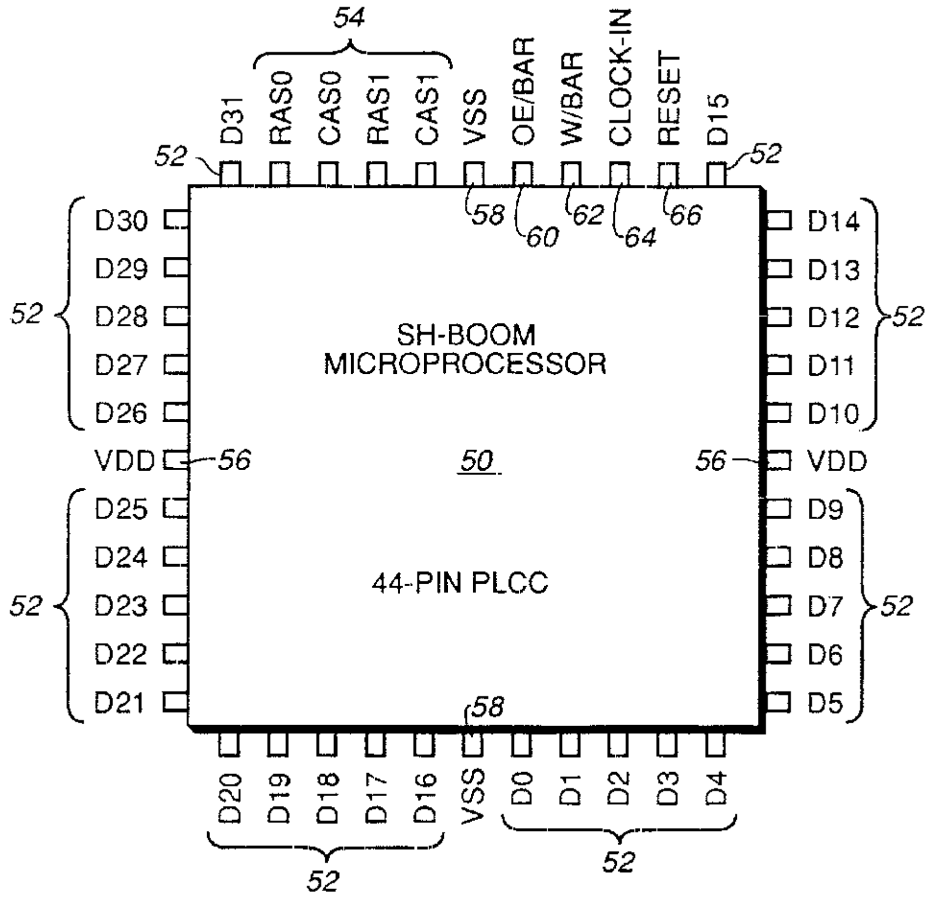
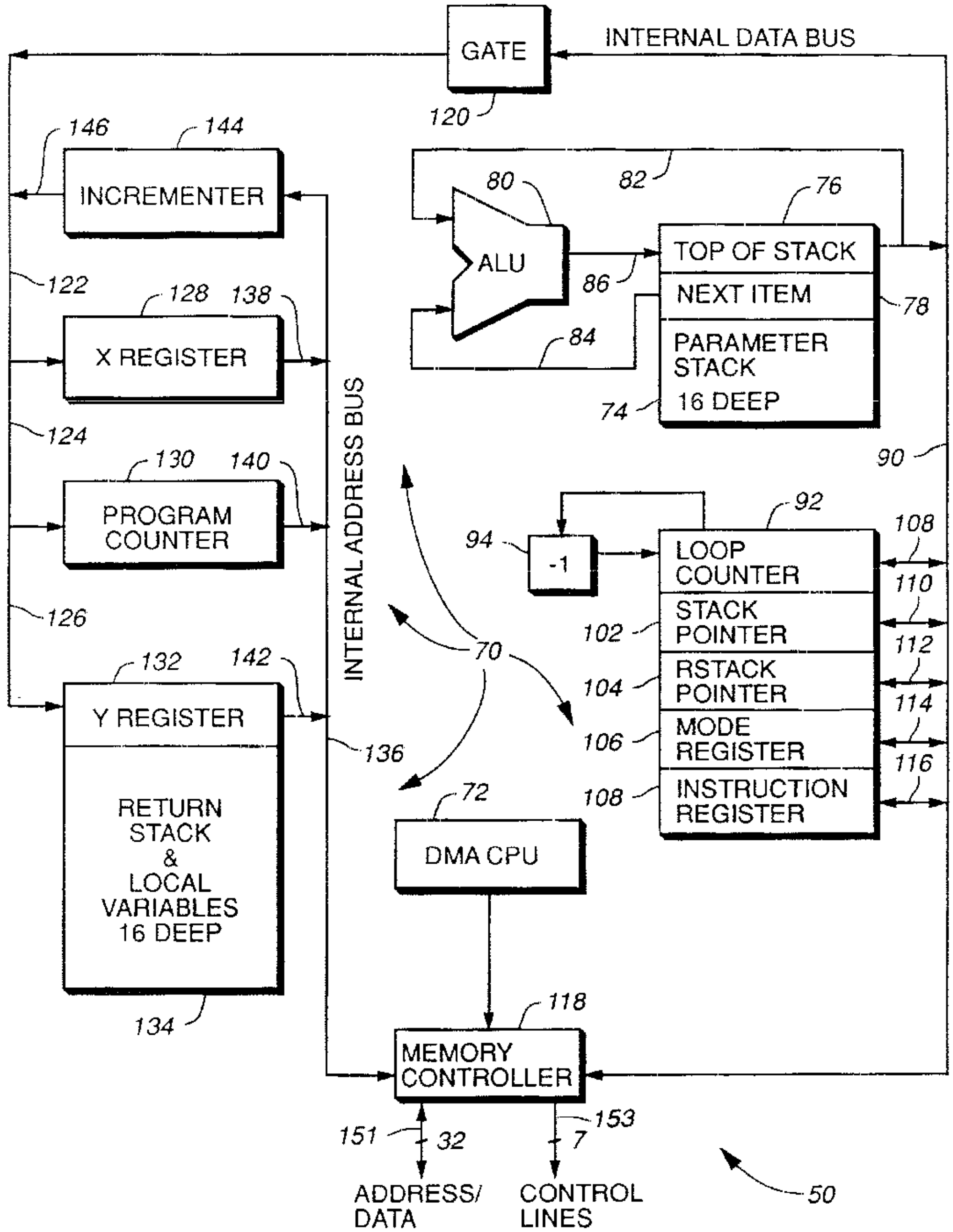

54
D31
RAS0
CAS0
RAS1
CAS1
VSS
OE/BAR
W/BAR
CLOCK-IN
RESET
D15
52
52
58
60
62
64
66
D30
D29
D28
D27
D26
52
D14
D13
D12
D11
D10
52
SH-BOOM
MICROPROCESSOR
VDD
56
50
56
VDD
D25
D24
D23
D22
D21
52
D9
D8
D7
D6
D5
52
44-PIN PLCC
58
D20
D19
D18
D17
D16
VSS
D0
D1
D2
D3
D4
52
52

FIG._1



GATE
INTERNAL DATA BUS
120
146
144
INCREMENTER
80
82
76
ALU
TOP OF STACK
NEXT ITEM
86
78
84
PARAMETER STACK
16 DEEP
74
122
128
138
X REGISTER
124
130
140
PROGRAM COUNTER
INTERNAL ADDRESS BUS
90
92
94
-1
LOOP COUNTER
108
110
STACK POINTER
102
112
RSTACK POINTER
104
114
MODE REGISTER
106
116
INSTRUCTION REGISTER
108
126
132
142
Y REGISTER
RETURN STACK & LOCAL VARIABLES 16 DEEP
70
136
72
DMA CPU
134
118
MEMORY CONTROLLER
151
32
153
7
ADDRESS/ DATA
CONTROL LINES
50

FIG._2

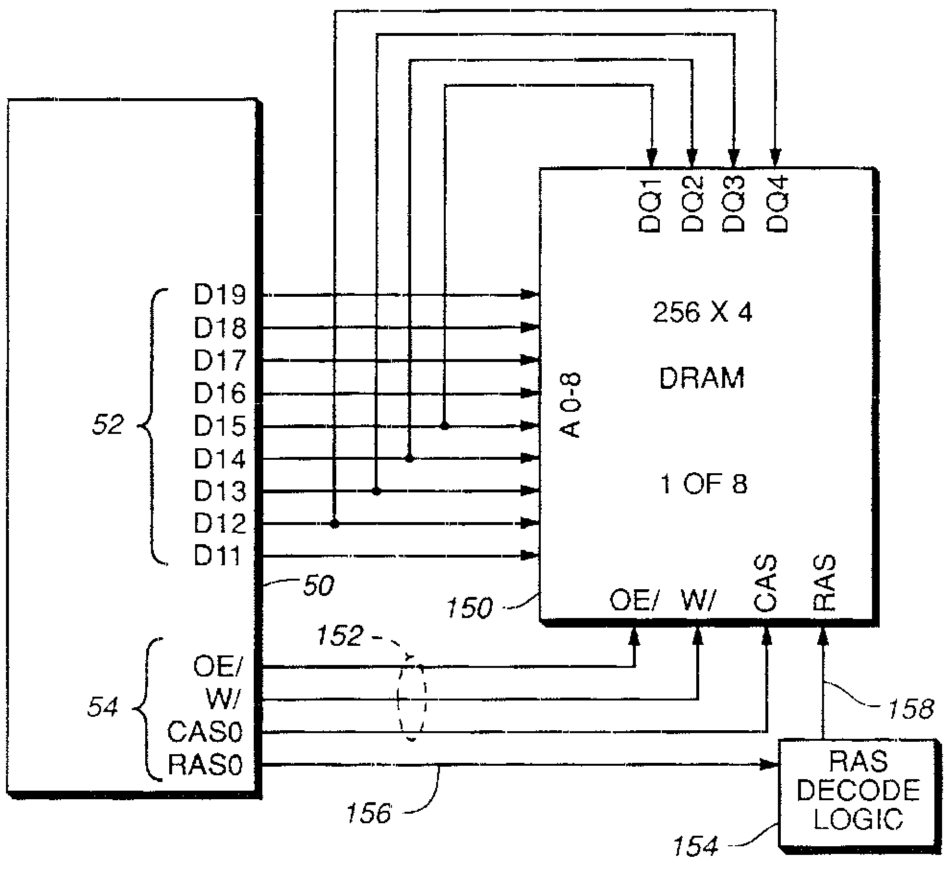

DQ1
DQ2
DQ3
DQ4
256 X 4
DRAM
1 OF 8
A 0-8
D19
D18
D17
D16
D15
D14
D13
D12
D11
52
50
150
OE/
W/
CAS
RAS
152
54
OE/
W/
CAS0
RAS0
158
RAS
DECODE
LOGIC
156
154

FIG._3

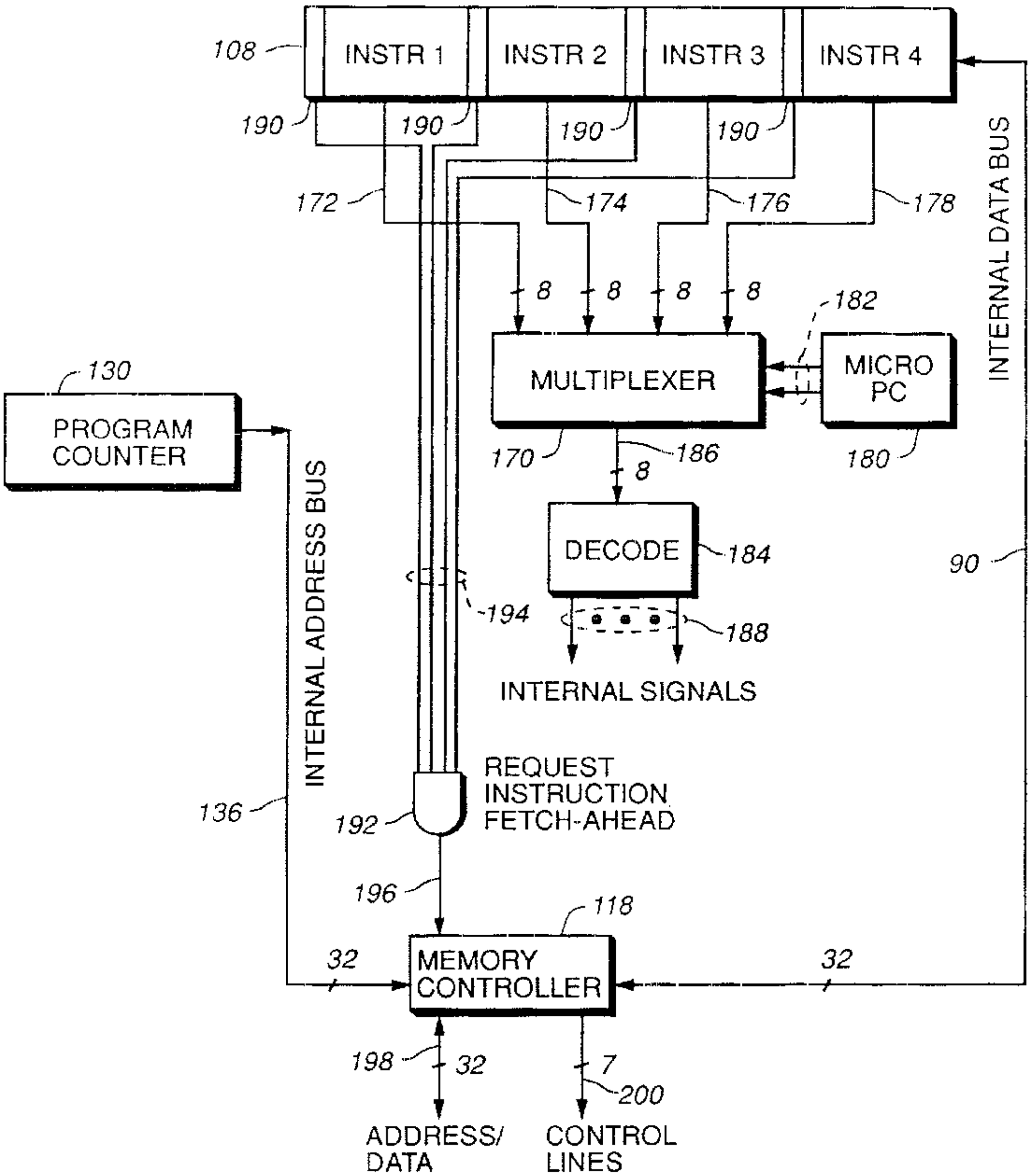
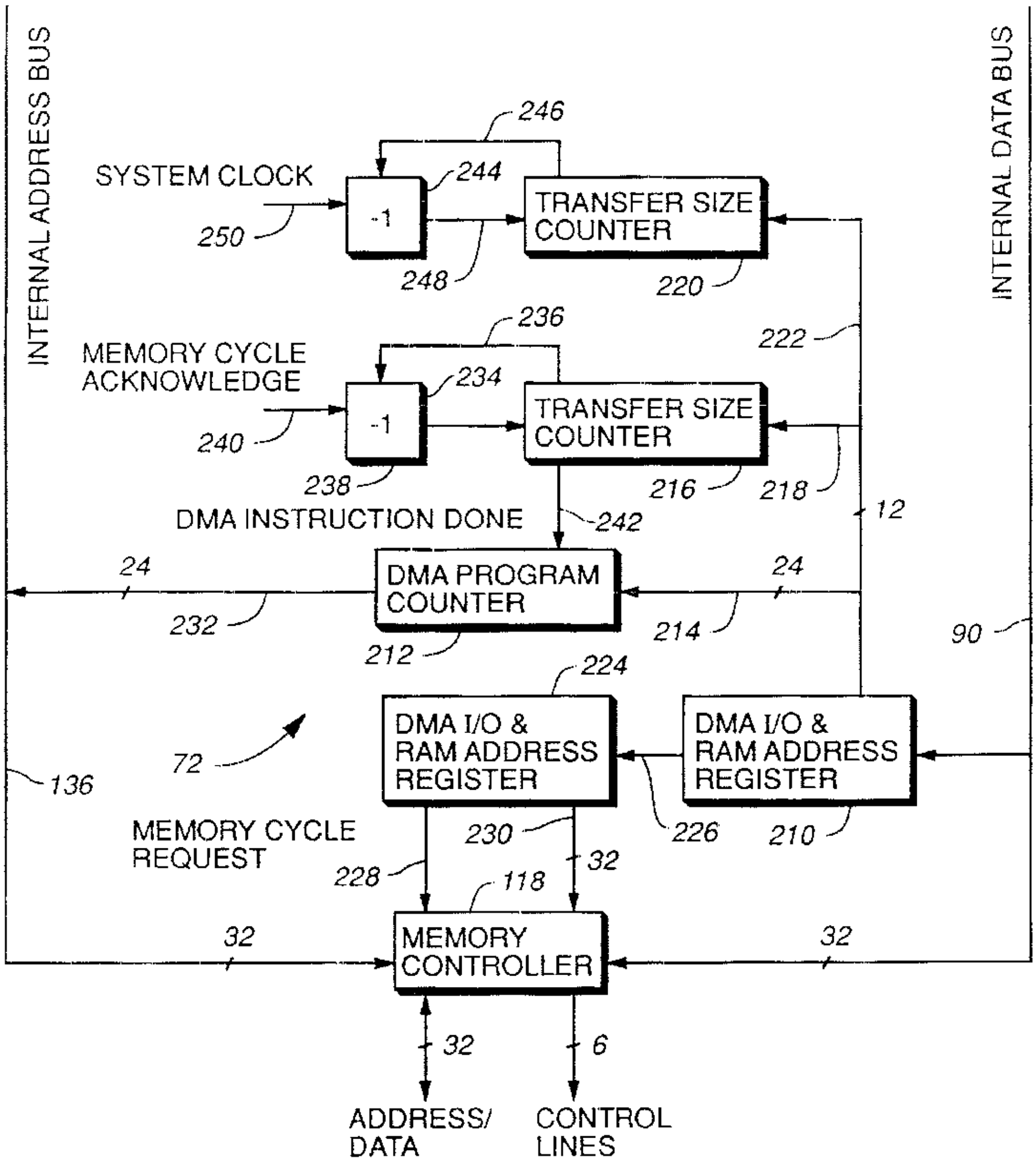
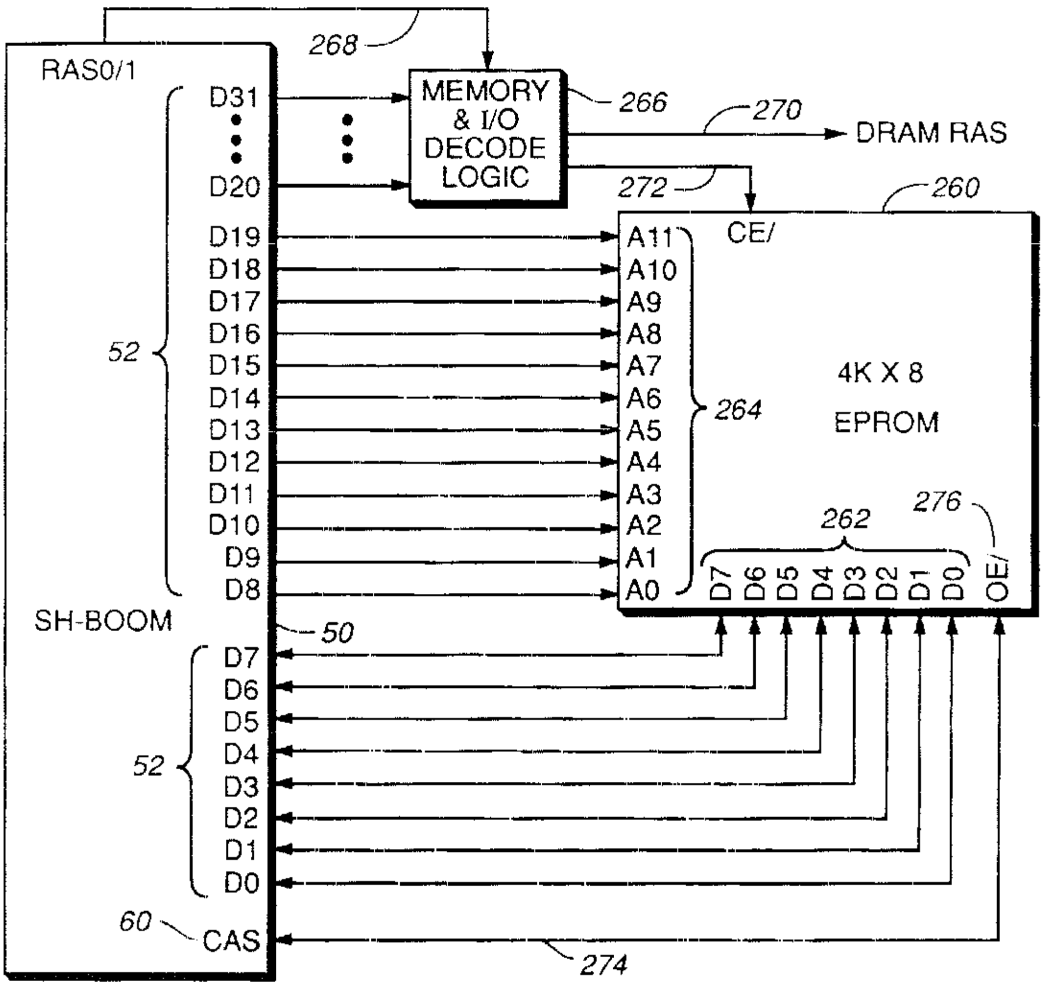

108
INSTR 1
INSTR 2
INSTR 3
INSTR 4
190
190
190
190
172
174
176
178
INTERNAL DATA BUS
8
8
8
8
182
MULTIPLEXER
MICRO PC
130
PROGRAM COUNTER
170
186
8
180
INTERNAL ADDRESS BUS
DECODE
184
90
194
188
INTERNAL SIGNALS
REQUEST INSTRUCTION FETCH-AHEAD
136
192
196
118
32
MEMORY CONTROLLER
32
198
32
7
200
ADDRESS/ DATA
CONTROL LINES

FIG._4



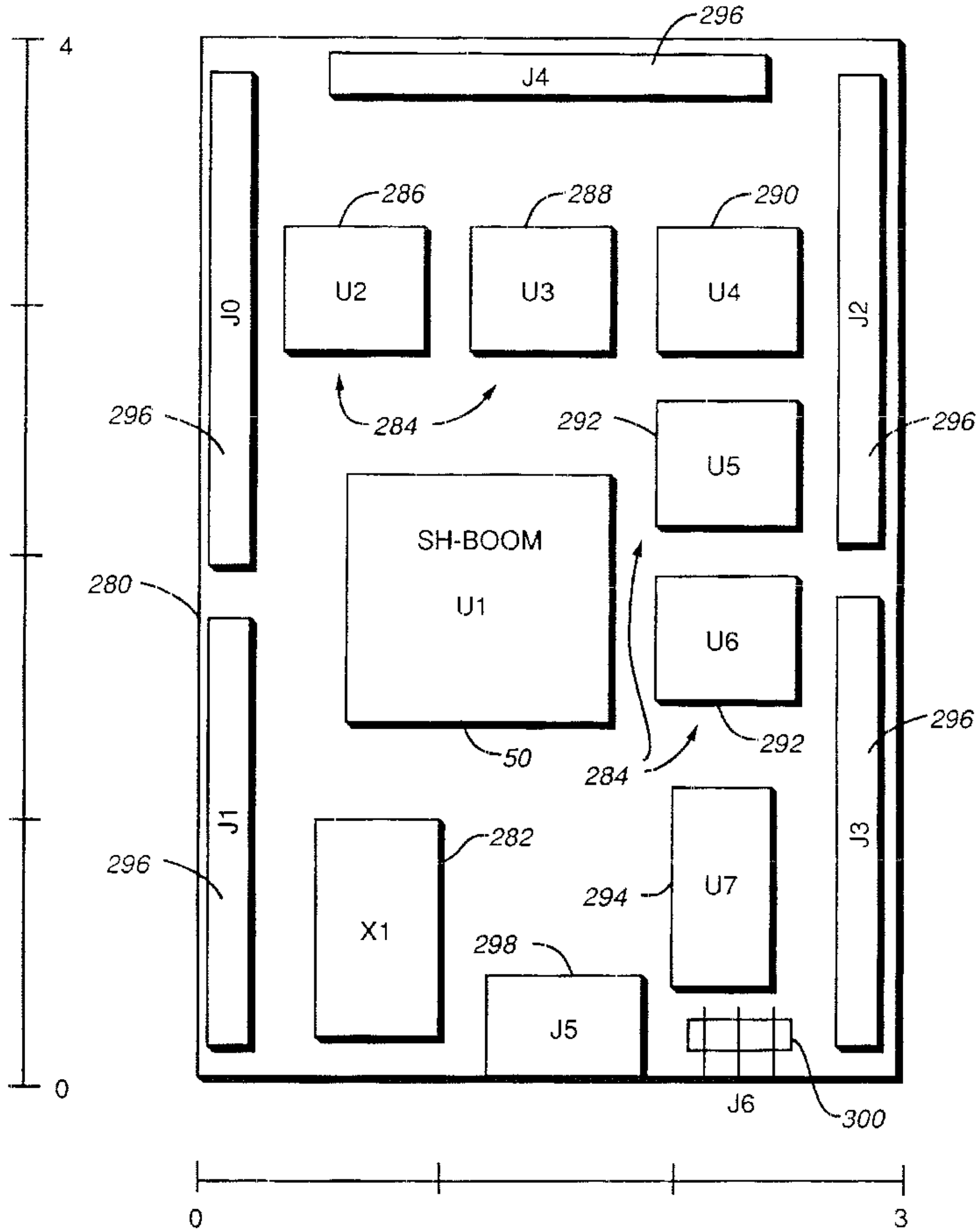
INTERNAL ADDRESS BUS
INTERNAL DATA BUS
246
SYSTEM CLOCK
244
-1
TRANSFER SIZE COUNTER
250
248
220
236
222
MEMORY CYCLE ACKNOWLEDGE
234
-1
TRANSFER SIZE COUNTER
240
238
216
218
12
DMA INSTRUCTION DONE
242
24
DMA PROGRAM COUNTER
24
232
214
90
212
224
DMA I/O & RAM ADDRESS REGISTER
DMA I/O & RAM ADDRESS REGISTER
72
136
MEMORY CYCLE REQUEST
230
226
210
32
228
118
32
MEMORY CONTROLLER
32
32
6
ADDRESS/ DATA
CONTROL LINES

FIG._5



RAS0/1
268
MEMORY & I/O DECODE LOGIC
266
270
DRAM RAS
272
260
CE/
D31
D20
D19
D18
D17
D16
D15
D14
D13
D12
D11
D10
D9
D8
52
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
264
4K X 8
EPROM
262
276
D7 D6 D5 D4 D3 D2 D1 D0 OE/
SH-BOOM
50
D7
D6
D5
D4
D3
D2
D1
D0
52
60
CAS
274

FIG._6



4
0
296
J4
J0
J2
286
288
290
U2
U3
U4
296
284
292
U5
296
SH-BOOM
U1
280
U6
50
284
292
296
J1
J3
282
296
X1
294
U7
298
J5
J6
300
0
3

FIG._7

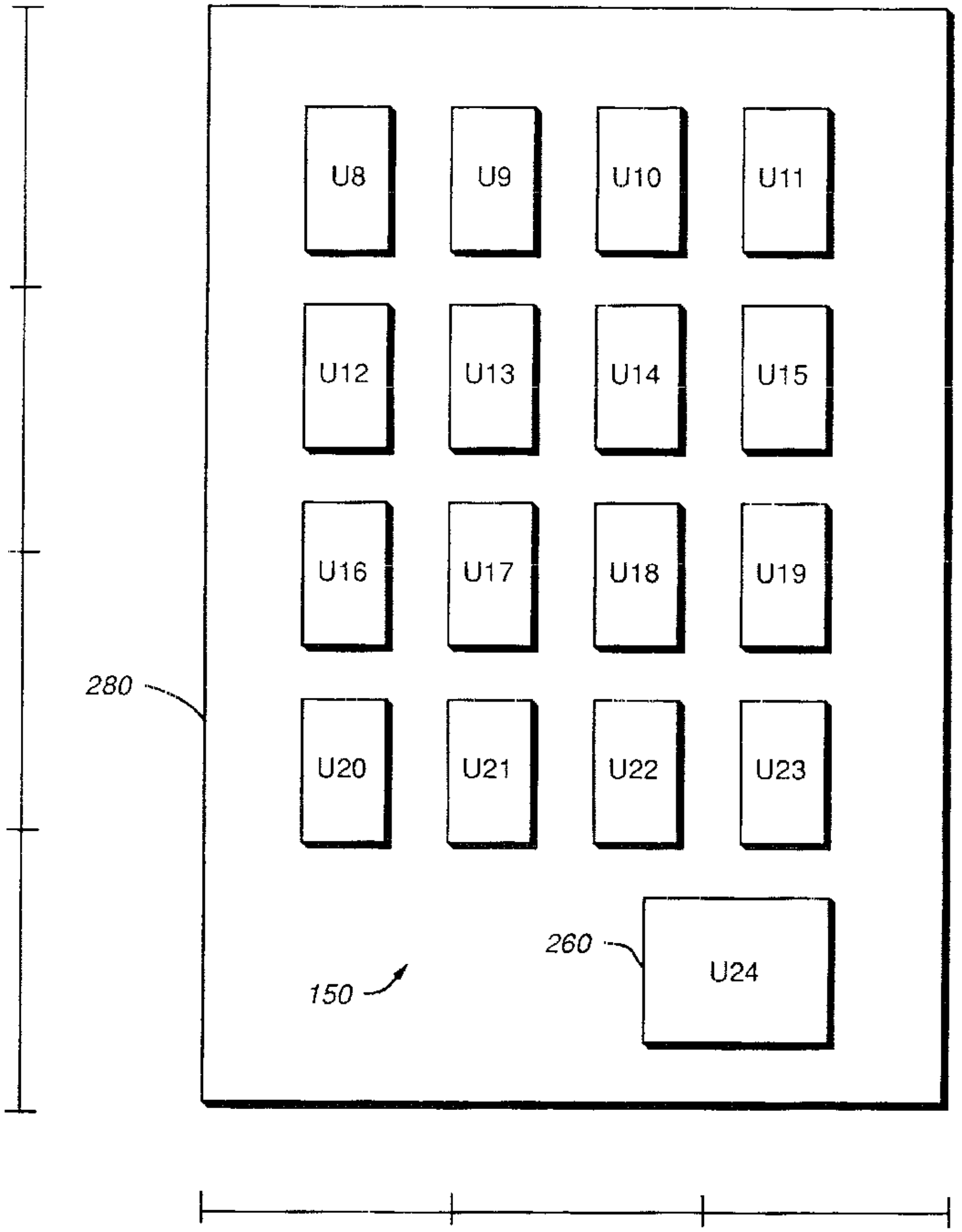

U8
U9
U10
U11
U12
U13
U14
U15
U16
U17
U18
U19
U20
U21
U22
U23
U24
280
260
150

FIG._8

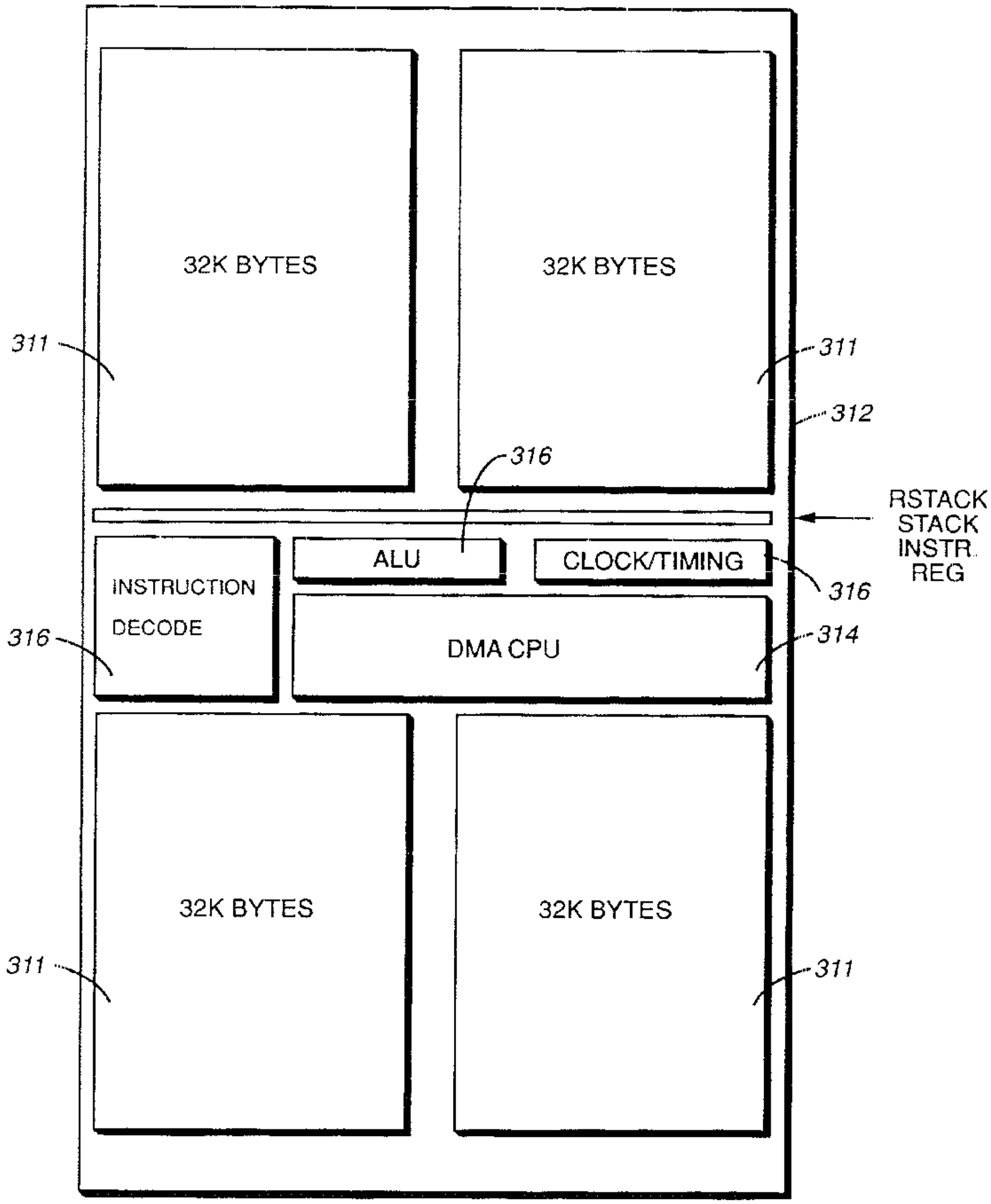

32K BYTES
32K BYTES
311
311
312
316
RSTACK
STACK
INSTR.
REG
ALU
CLOCK/TIMING
INSTRUCTION
DECODE
316
316
DMA CPU
314
32K BYTES
32K BYTES
311
311

FIG._9

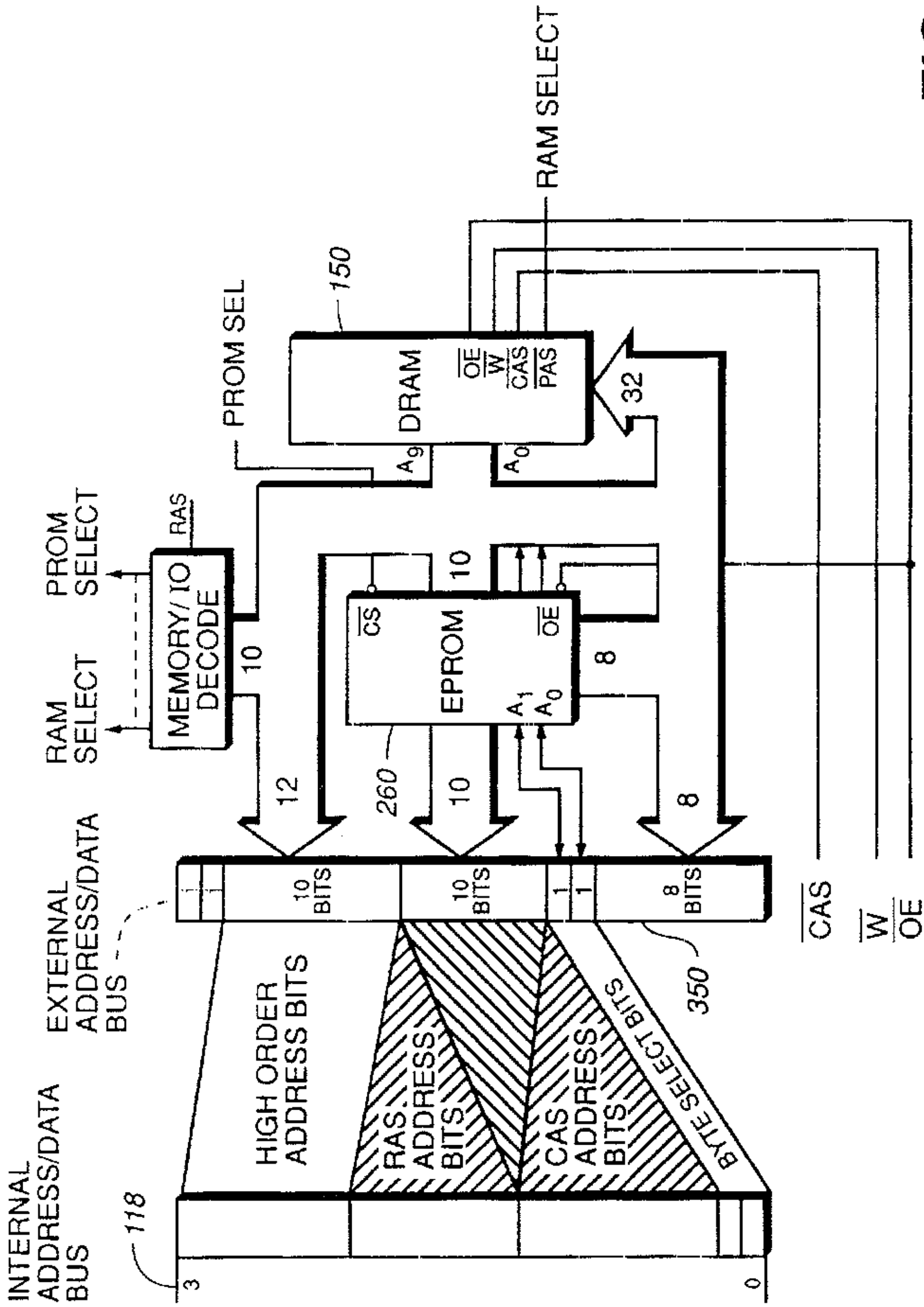
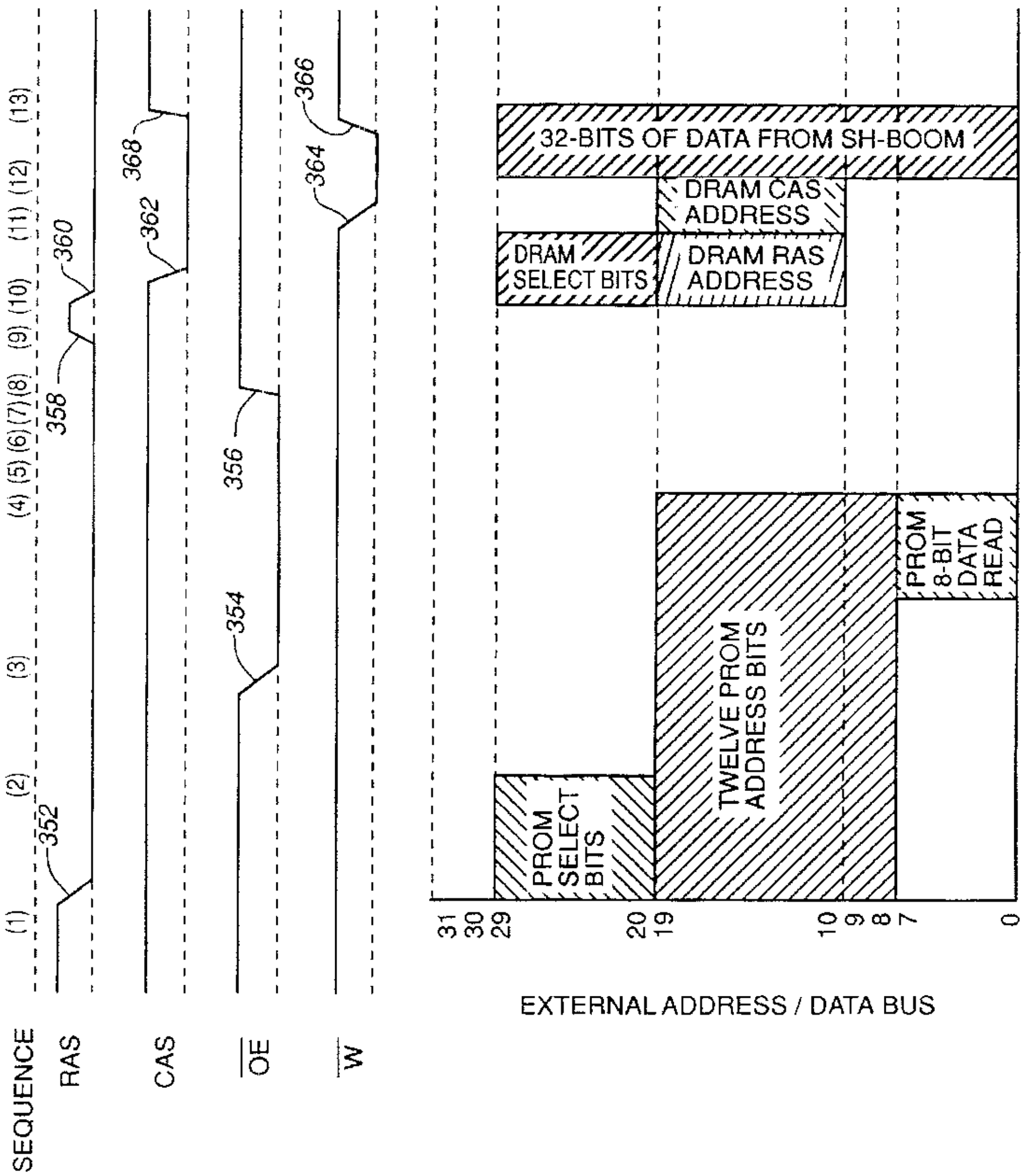

RAM SELECT
150
PROM SEL
DRAM
OE
W
CAS
PAS
32
A9
A0
RAS
PROM SELECT
RAM SELECT
MEMORY/IO DECODE
10
10
CS
EPROM
OE
A1
A0
8
12
260
10
8
EXTERNAL ADDRESS/DATA BUS
10 BITS
10 BITS
1
1
8 BITS
CAS
W
OE
HIGH ORDER ADDRESS BITS
RAS ADDRESS BITS
CAS ADDRESS BITS
BYTE SELECT BITS
350
INTERNAL ADDRESS/DATA BUS
118
3
0

FIG._10

FIG._11


SEQUENCE
RAS
CAS
OE
W
(1)
(2)
(3)
(4) (5) (6) (7) (8)
(9) (10)
(11) (12)
(13)
352
354
356
358
360
362
364
366
368
32-BITS OF DATA FROM SH-BOOM
DRAM CAS ADDRESS
DRAM SELECT BITS
DRAM RAS ADDRESS
PROM SELECT BITS
TWELVE PROM ADDRESS BITS
PROM 8-BIT DATA READ
31
30
29
20
19
10
9
8
7
0
EXTERNAL ADDRESS / DATA BUS

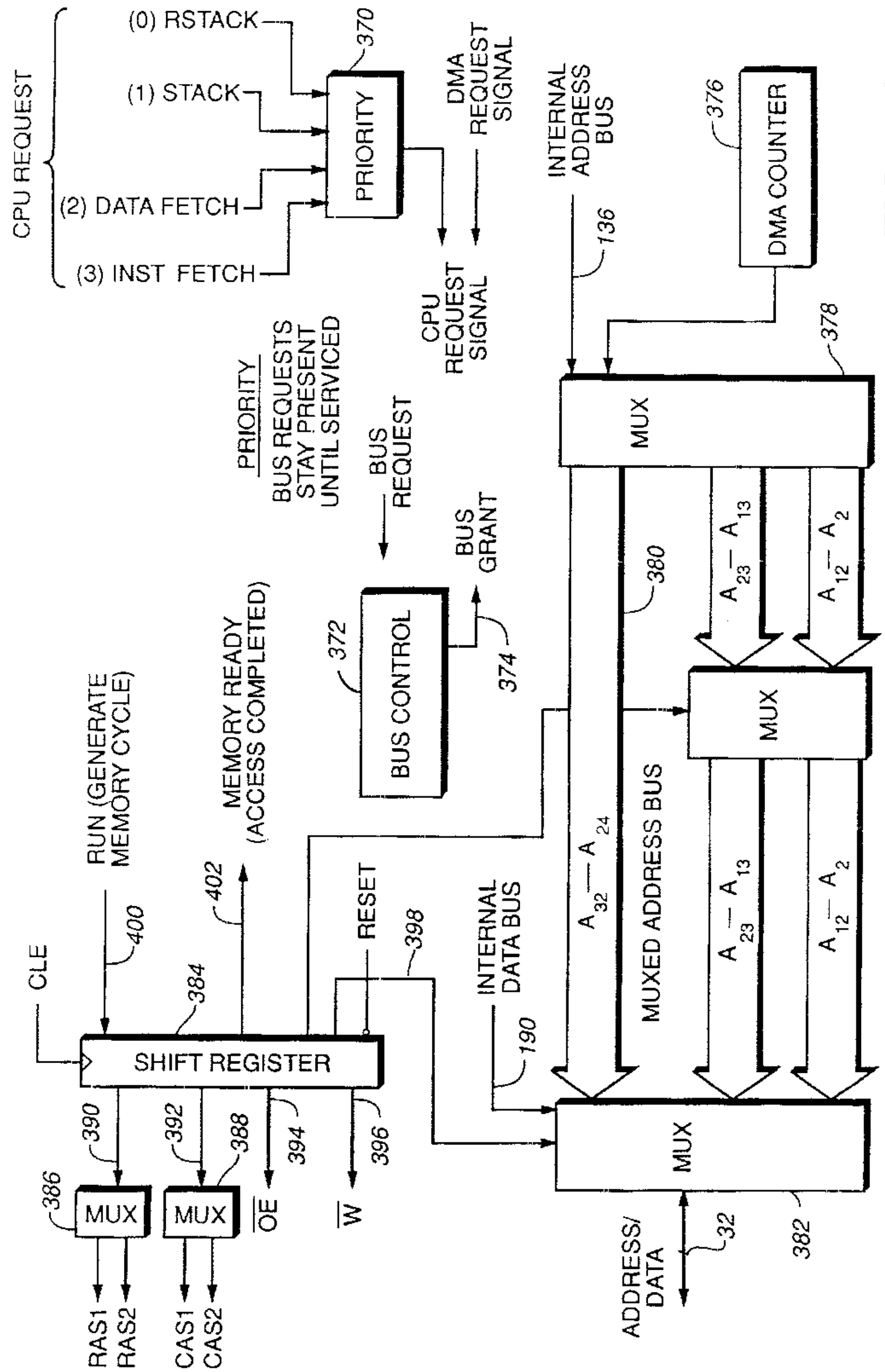
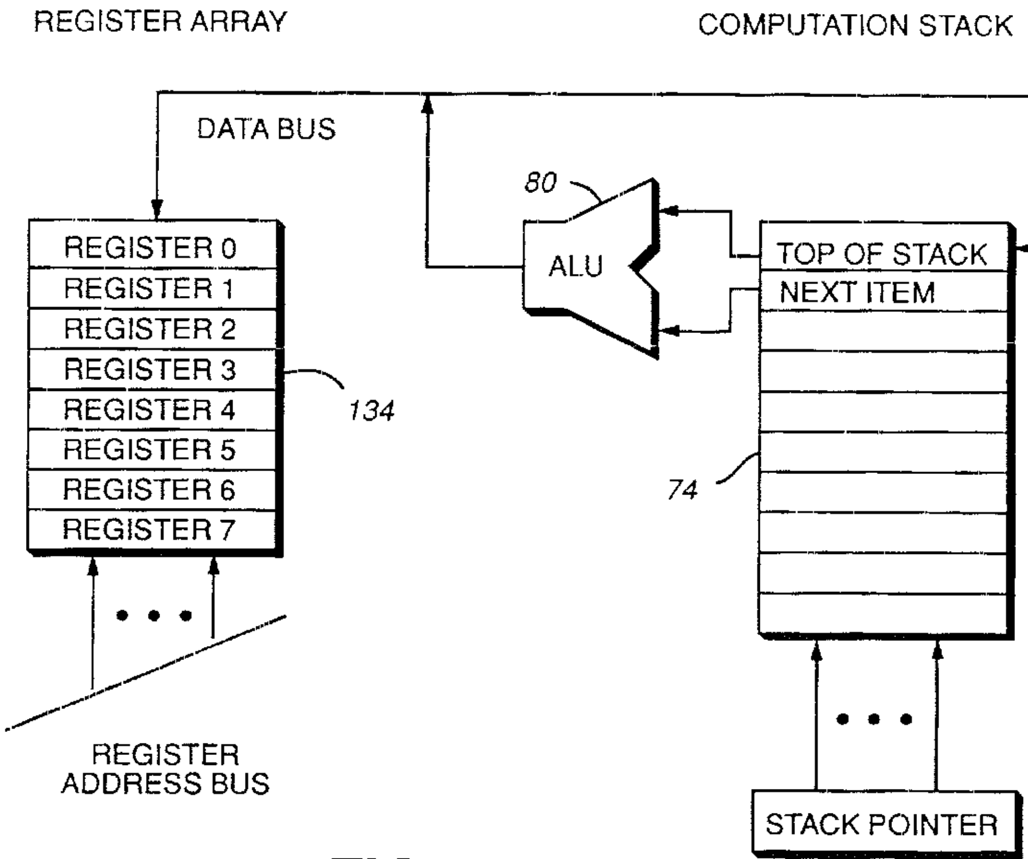

CPU REQUEST
(0) RSTACK
(1) STACK
(2) DATA FETCH
(3) INST FETCH
370
PRIORITY
DMA REQUEST SIGNAL
CPU REQUEST SIGNAL
INTERNAL ADDRESS BUS
136
376
DMA COUNTER
378
MUX
PRIORITY
BUS REQUESTS STAY PRESENT UNTIL SERVICED
BUS REQUEST
BUS GRANT
372
BUS CONTROL
374
380
A_{23} — A_{13}
A_{12} — A_2
MUX
RUN (GENERATE MEMORY CYCLE)
MEMORY READY (ACCESS COMPLETED)
CLE
400
402
384
SHIFT REGISTER
RESET
398
INTERNAL DATA BUS
190
A_{32} — A_{24}
MUXED ADDRESS BUS
A_{23} — A_{13}
A_{12} — A_2
390
392
388
394
396
386
MUX
MUX
$\overline{OE}$
$\overline{W}$
RAS1
RAS2
CAS1
CAS2
MUX
ADDRESS/DATA
32
382

FIG._12



REGISTER ARRAY
COMPUTATION STACK
DATA BUS
80
ALU
REGISTER 0
REGISTER 1
REGISTER 2
REGISTER 3
REGISTER 4
REGISTER 5
REGISTER 6
REGISTER 7
134
TOP OF STACK
NEXT ITEM
74
REGISTER
ADDRESS BUS
STACK POINTER

FIG._13

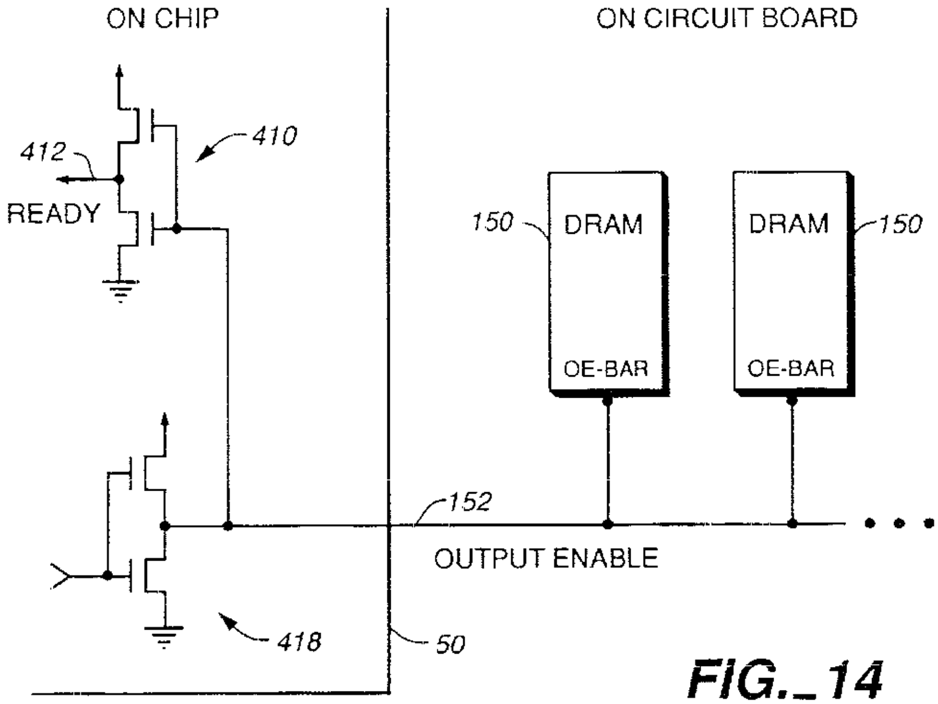

ON CHIP
ON CIRCUIT BOARD
412
410
READY
150
DRAM
DRAM
150
OE-BAR
OE-BAR
152
OUTPUT ENABLE
418
50

FIG._14

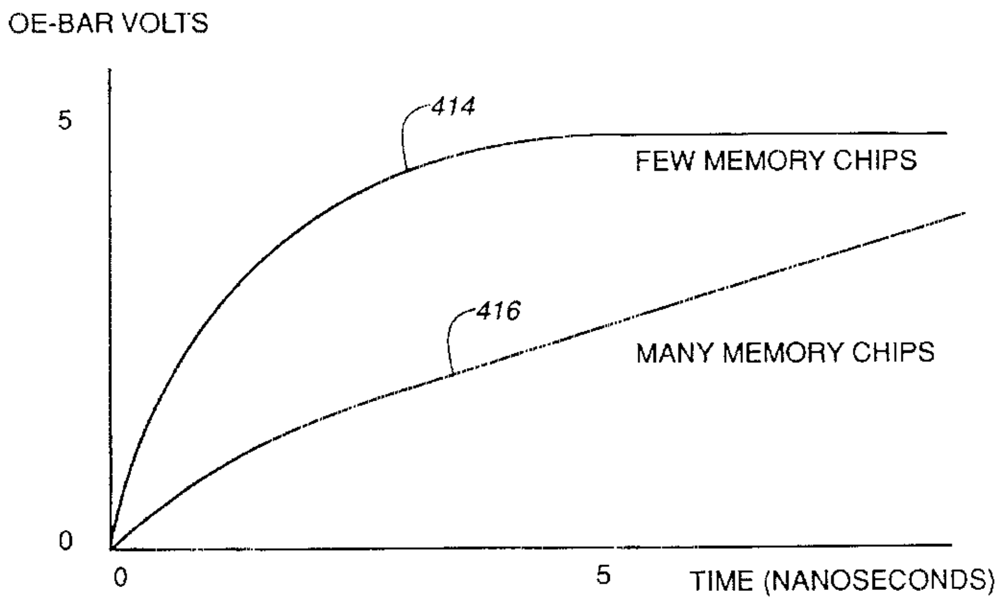
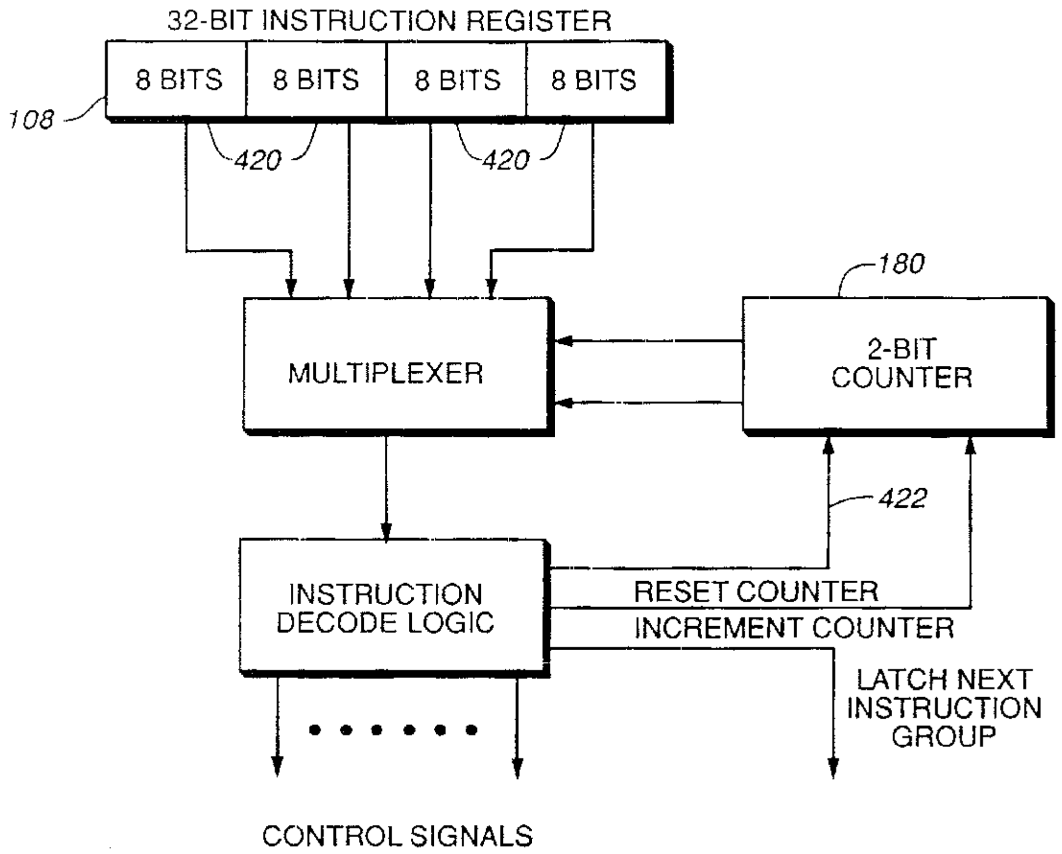
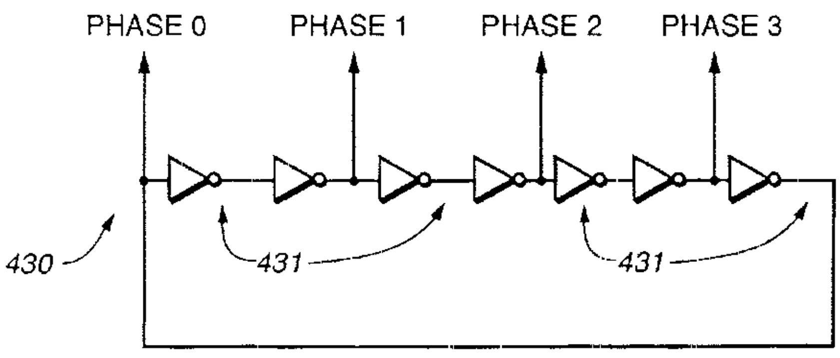

OE-BAR VOLTS
5
414
FEW MEMORY CHIPS
416
MANY MEMORY CHIPS
0
0
5
TIME (NANOSECONDS)

FIG._15



32-BIT INSTRUCTION REGISTER
8 BITS
8 BITS
8 BITS
8 BITS
108
420
420
180
MULTIPLEXER
2-BIT COUNTER
422
INSTRUCTION DECODE LOGIC
RESET COUNTER
INCREMENT COUNTER
LATCH NEXT INSTRUCTION GROUP
CONTROL SIGNALS

FIG._16



PHASE 0
PHASE 1
PHASE 2
PHASE 3
430
431
431

FIG._18

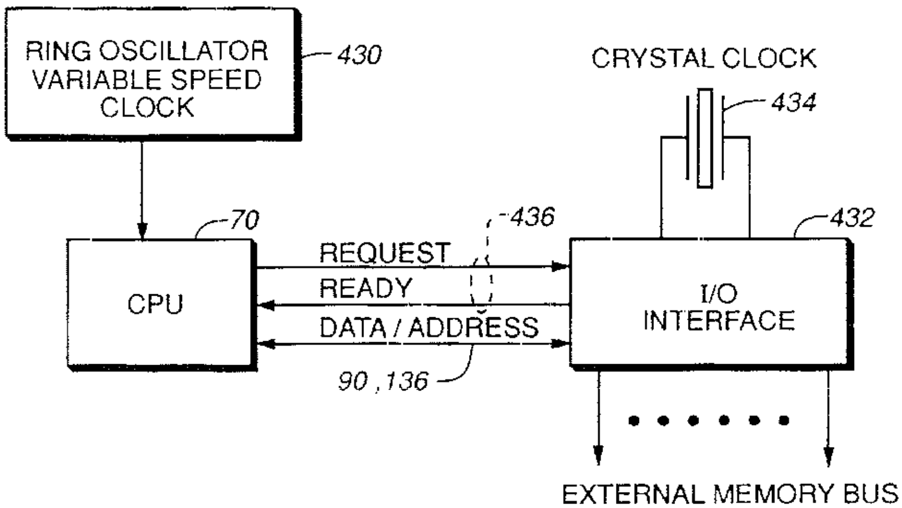

RING OSCILLATOR VARIABLE SPEED CLOCK
430
CRYSTAL CLOCK
434
70
436
432
CPU
REQUEST
READY
DATA / ADDRESS
I/O INTERFACE
90 ,136
EXTERNAL MEMORY BUS

FIG._17

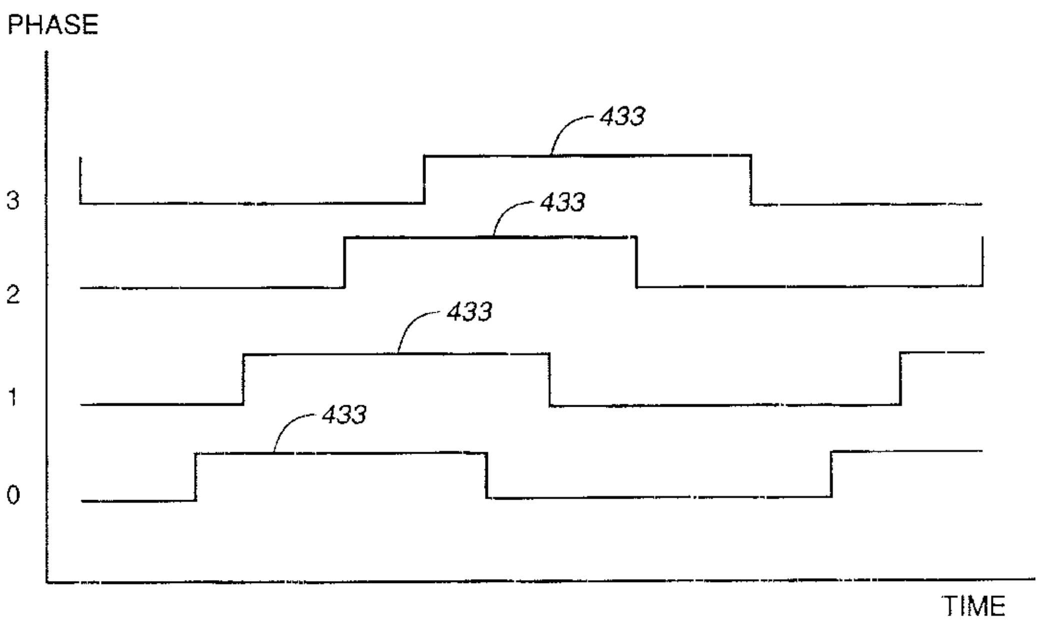
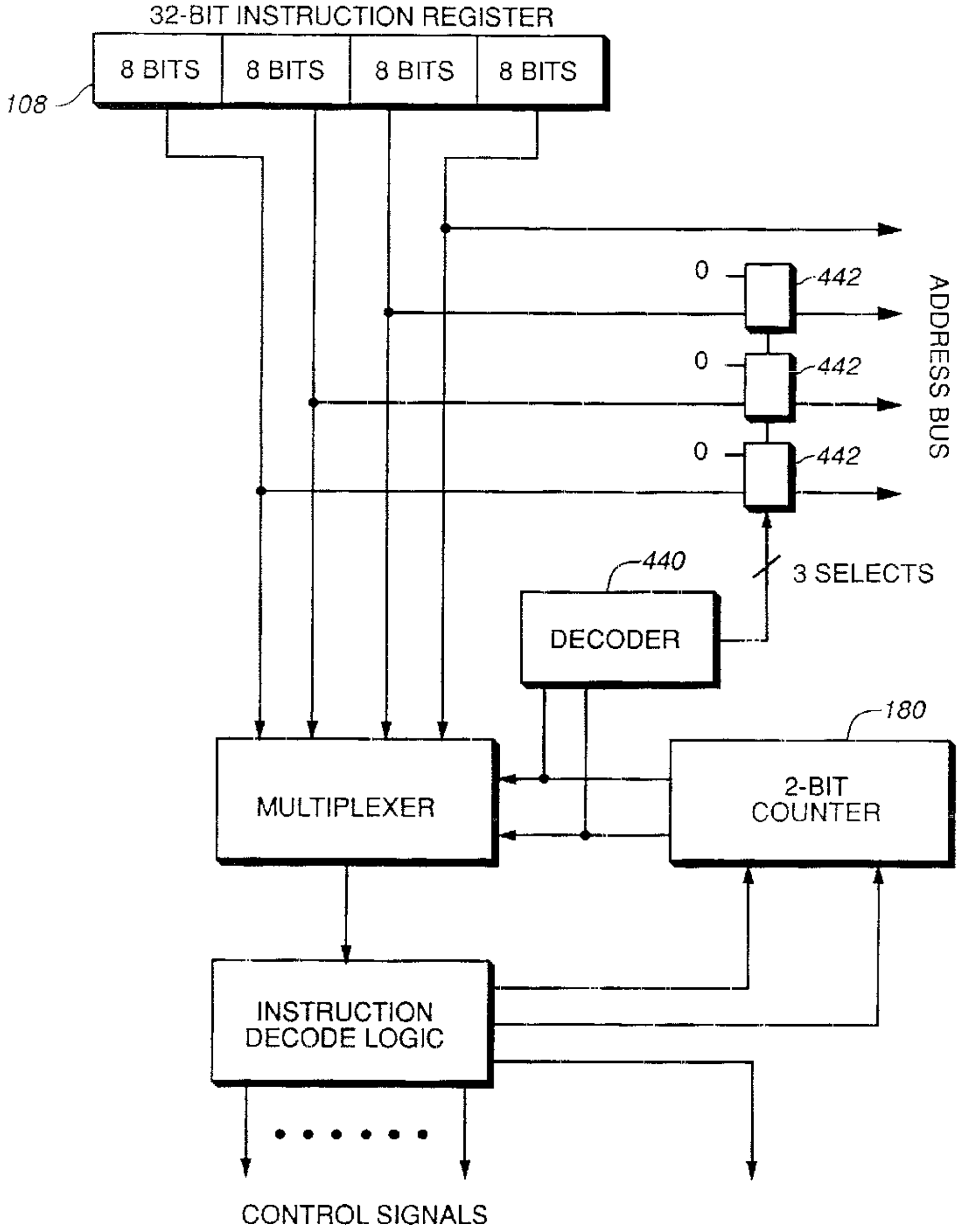
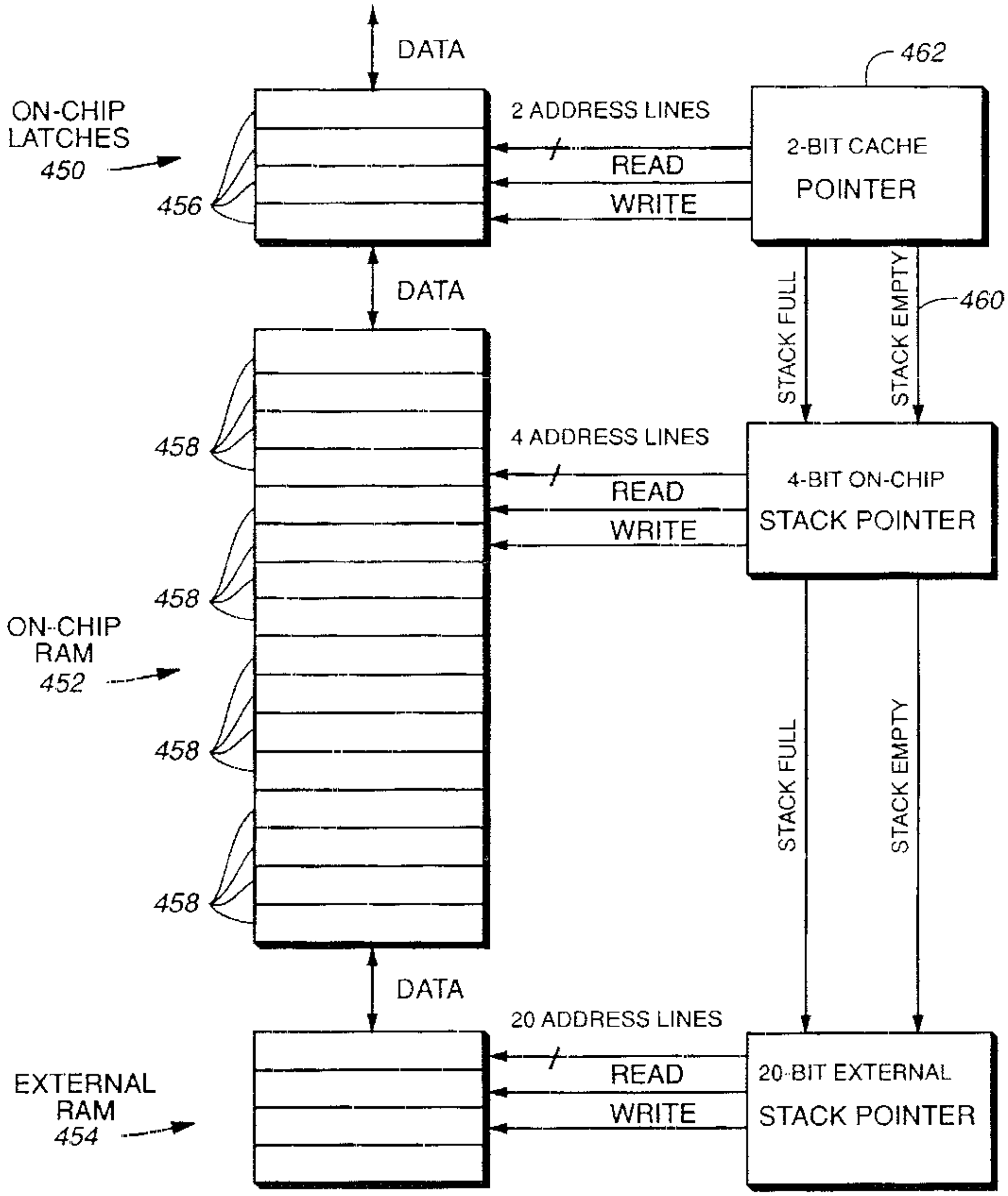

PHASE
433
3
2
1
0
TIME

FIG._19



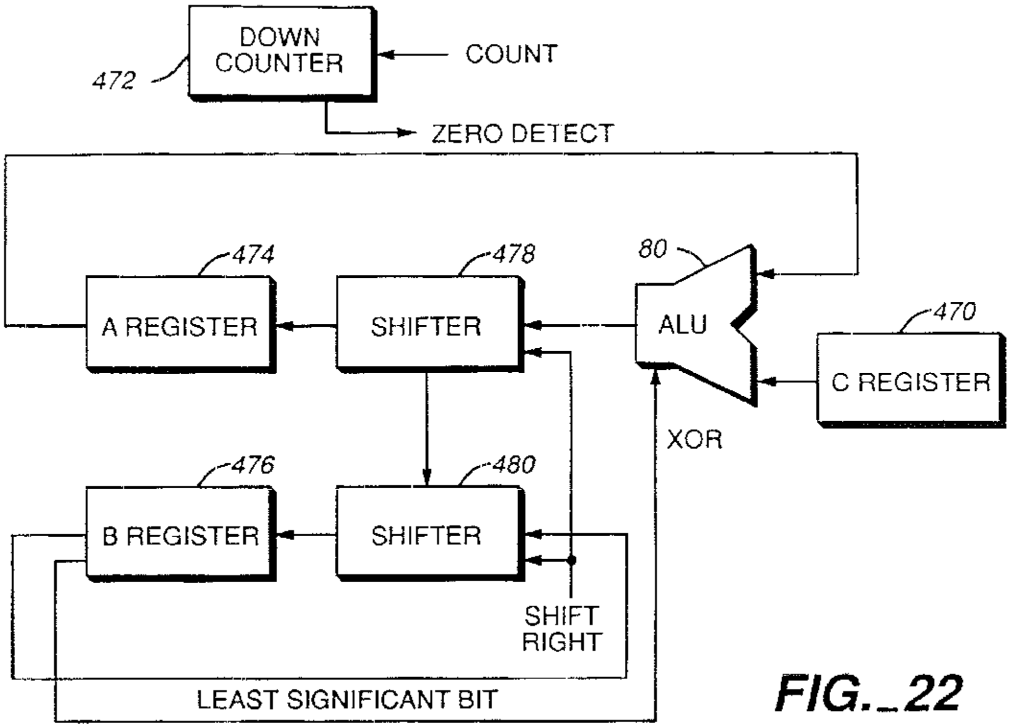
32-BIT INSTRUCTION REGISTER
8 BITS
8 BITS
8 BITS
8 BITS
108
0
442
0
442
0
442
ADDRESS BUS
440
3 SELECTS
DECODER
180
MULTIPLEXER
2-BIT
COUNTER
INSTRUCTION
DECODE LOGIC
CONTROL SIGNALS

FIG._20



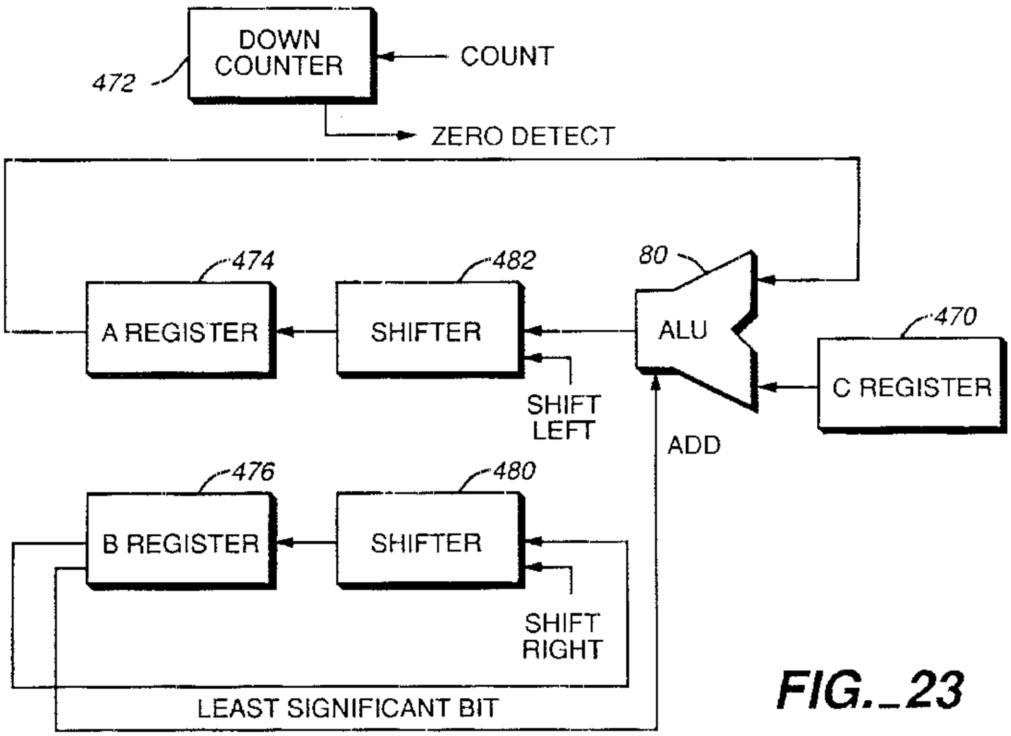
DATA
462
ON-CHIP
LATCHES
450
2 ADDRESS LINES
2-BIT CACHE
POINTER
READ
WRITE
456
DATA
STACK FULL
STACK EMPTY
460
458
4 ADDRESS LINES
4-BIT ON-CHIP
STACK POINTER
READ
WRITE
458
ON-CHIP
RAM
452
458
STACK FULL
STACK EMPTY
458
DATA
20 ADDRESS LINES
EXTERNAL
RAM
454
READ
WRITE
20-BIT EXTERNAL
STACK POINTER

FIG._21



DOWN COUNTER
472
COUNT
ZERO DETECT
474
478
80
A REGISTER
SHIFTER
ALU
470
C REGISTER
XOR
476
480
B REGISTER
SHIFTER
SHIFT RIGHT
LEAST SIGNIFICANT BIT

FIG._22



DOWN COUNTER
472
COUNT
ZERO DETECT
474
482
80
A REGISTER
SHIFTER
ALU
470
C REGISTER
SHIFT LEFT
ADD
476
480
B REGISTER
SHIFTER
SHIFT RIGHT
LEAST SIGNIFICANT BIT

FIG._23

HIGH PERFORMANCE MICROPROCESSOR HAVING VARIABLE SPEED SYSTEM CLOCK

CROSS REFERENCE TO RELATED APPLICATIONS

This application is a division of U.S. application Ser. No. 07/389,334, filed Aug. 3, 1989, now U.S. Pat. No. 5,440,749.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to a simplified, reduced instruction set computer (RISC) microprocessor. More particularly, it relates to such a microprocessor which is capable of performance levels of, for example, 20 million instructions per second (MIPS) at a price of, for example, 20 dollars.

2. Description of the Prior Art

Since the invention of the microprocessor, improvements in its design have taken two different approaches. In the first approach, a brute force gain in performance has been achieved through the provision of greater numbers of faster transistors in the microprocessor integrated circuit and an instruction set of increased complexity. This approach is exemplified by the Motorola 68000 and Intel 80X86 microprocessor families. The trend in this approach is to larger die sizes and packages, with hundreds of pinouts.

More recently, it has been perceived that performance gains can be achieved through comparative simplicity, both in the microprocessor integrated circuit itself and in its instruction set. This second approach provides RISC microprocessors, and is exemplified by the Sun SPARC and the Intel 8960 microprocessors. However, even with this approach as conventionally practiced, the packages for the microprocessor are large, in order to accommodate the large number of pinouts that continue to be employed. A need therefore remains for further simplification of high performance microprocessors.

With conventional high performance microprocessors, fast static memories are required for direct connection to the microprocessors in order to allow memory accesses that are fast enough to keep up with the microprocessors. Slower dynamic random access memories (DRAMs) are used with such microprocessors only in a hierarchical memory arrangement, with the static memories acting as a buffer between the microprocessors and the DRAMs. The necessity to use static memories increases cost of the resulting systems.

Conventional microprocessors provide direct memory accesses (DMA) for system peripheral units through DMA controllers, which may be located on the microprocessor integrated circuit, or provided separately. Such DMA controllers can provide routine handling of DMA requests and responses, but some processing by the main central processing unit (CPU) of the microprocessor is required.

SUMMARY OF THE INVENTION

Accordingly, it is an object of this invention to provide a microprocessor with a reduced pin count and cost compared to conventional microprocessors.

It is another object of the invention to provide a high performance microprocessor that can be directly connected to DRAMs without sacrificing microprocessor speed.

It is a further object of the invention to provide a high performance microprocessor in which DMA does not require use of the main CPU during DMA requests and responses and which provides very rapid DMA response with predictable response times.

The attainment of these and related objects may be achieved through use of the novel high performance, low cost microprocessor herein disclosed. In accordance with one aspect of the invention, a microprocessor system in accordance with this invention has a central processing unit, a dynamic random access memory and a bus connecting the central processing unit to the dynamic random access memory. There is a multiplexing means on the bus between the central processing unit and the dynamic random access memory. The multiplexing means is connected and configured to provide row addresses, column addresses and data on the bus.

In accordance with another aspect of the invention, the microprocessor system has a means connected to the bus for fetching instructions for the central processing unit on the bus. The means for fetching instructions is configured to fetch multiple sequential instructions in a single memory cycle. In a variation of this aspect of the invention, a programmable read only memory containing instructions for the central processing unit is connected to the bus. The means for fetching instructions includes means for assembling a plurality of instructions from the programmable read only memory and storing the plurality of instructions in the dynamic random access memory.

In another aspect of the invention, the microprocessor system includes a central processing unit, a direct memory access processing unit and a memory connected by a bus. The direct memory access processing unit includes means for fetching instructions for the central processing unit and for fetching instructions for the direct memory access processing unit on the bus.

In a further aspect of the invention, the microprocessor system, including the memory, is contained in an integrated circuit. The memory is a dynamic random access memory, and the means for fetching multiple instructions includes a column latch for receiving the multiple instructions.

In still another aspect of the invention, the microprocessor system additionally includes an instruction register for the multiple instructions connected to the means for fetching instructions. A means is connected to the instruction register for supplying the multiple instructions in succession from the instruction register. A counter is connected to control the means for supplying the multiple instructions to supply the multiple instructions in succession. A means for decoding the multiple instructions is connected to receive the multiple instructions in succession from the means for supplying the multiple instructions. The counter is connected to said means for decoding to receive incrementing and reset control signals from the means for decoding. The means for decoding is configured to supply the reset control signal to the counter and to supply a control signal to the means for fetching instructions in response to a SKIP instruction in the multiple instructions. In a modification of this aspect of the invention, the microprocessor system additionally has a loop counter connected to receive a decrement control signal from the means for decoding. The means for decoding is configured to supply the reset control signal to the counter and the decrement control signal to the loop counter in response to a MICROLOOP instruction in the multiple instructions. In a further modification to this aspect of the invention, the means for decoding is configured to control

the counter in response to an instruction utilizing a variable width operand A means is connected to the counter to select the variable width operand in response to the counter

In a still further aspect of the invention, the microprocessor system includes an arithmetic logic unit A first push down stack is connected to the arithmetic logic unit The first push down stack includes means for storing a top item connected to a first input of the arithmetic logic unit and means for storing a next item connected to a second input of the arithmetic logic unit The arithmetic logic unit has an output connected to the means for storing a top item The means for storing a top item is connected to provide an input to a register file The register file desirably is a second push down stack, and the means for storing a top item and the register file are bidirectionally connected

In another aspect of the invention, a data processing system has a microprocessor including a sensing circuit and a driver circuit, a memory and an output enable line connected between the memory, the sensing circuit and the driver circuit The sensing circuit is configured to provide a ready signal when the output enable line reaches a predetermined electrical level, such as a voltage The microprocessor is configured so that the driver circuit provides an enabling signal on the output enable line responsive to the ready signal

In a further aspect of the invention, the microprocessor system has a ring counter variable speed system clock connected to the central processing unit The central processing unit and the ring counter variable speed system clock are provided in a single integrated circuit. An input/output interface is connected to exchange coupling control signals, addresses and data with the input/output interface A second clock independent of the ring counter variable speed system clock is connected to the input/output interface

In yet another aspect of the invention, a push down stack is connected to the arithmetic logic unit The push down stack includes means for storing a top item connected to a first input of the arithmetic logic unit and means for storing a next item connected to a second input of the arithmetic logic unit The arithmetic logic unit has an output connected to the means for storing a top item The push down stack has a first plurality of stack elements configured as latches and a second plurality of stack elements configured as a random access memory The first and second plurality of stack elements and the central processing unit are provided in a single integrated circuit A third plurality of stack elements is configured as a random access memory external to the single integrated circuit In this aspect of the invention, desirably a first pointer is connected to the first plurality of stack elements, a second pointer connected to the second plurality of stack elements, and a third pointer is connected to the third plurality of stack elements The central processing unit is connected to pop items from the first plurality of stack elements The first stack pointer is connected to the second stack pointer to pop a first plurality of items from the second plurality of stack elements when the first plurality of stack elements are empty from successive pop operations by the central processing unit The second stack pointer is connected to the third stack pointer to pop a second plurality of items from the third plurality of stack elements when the second plurality of stack elements are empty from successive pop operations by the central processing unit

In another aspect of the invention a first register is connected to supply a first input to the arithmetic logic unit A first shifter is connected between an output of the arithmetic logic unit and the first register A second register is connected to receive a starting polynomial value An output of the second register is connected to a second shifter A least significant bit of the second register is connected to The arithmetic logic unit A third register is connected to supply feedback terms of a polynomial to the arithmetic logic unit A down counter, for counting down a number corresponding to digits of a polynomial to be generated, is connected to the arithmetic logic unit The arithmetic logic unit is responsive to a polynomial instruction to carry out an exclusive OR of the contents of the first register with the contents of the third register if the least significant bit of the second register is a "ONE" and to pass the contents of the first register unaltered if the least significant bit of the second register is a "ZERO", until the down counter completes a count The polynomial to be generated results in said first register

In still another aspect of the invention, a result register is connected to supply a first input to the arithmetic logic unit. A first, left shifting shifter is connected between an output of the arithmetic logic unit and the result register A multiplier register is connected to receive a multiplier in bit reversed form An output of the multiplier register is connected to a second right shitting shifter A least significant bit of the multiplier register is connected to the arithmetic logic unit. A third register is connected to supply a multiplicand to said arithmetic logic unit A down counter, for counting down a number corresponding to one less than the number of digits of the multiplier, is connected to the arithmetic logic unit The arithmetic logic unit is responsive to a multiply instruction to add the contents of the result register with the contents of the third register, when the least significant bit of the multiplier register is a "ONE" and to pass the contents of the result register unaltered, until the down counter completes a count The product results in the result register

The attainment of the foregoing and related objects advantages and features of the invention should be more readily apparent to those skilled in the art, after review of the following more detailed description of the invention, taken together with the drawings, in which:

BRIEF DESCRIPTION OF THE DRAWINGS

FIG 1 is an external plan view of an integrated circuit package incorporating a microprocessor in accordance with the invention

FIG 2 is a block diagram of a microprocessor in accordance with the invention

FIG 3 is a block diagram of a portion of a data processing system incorporating the microprocessor of FIGS 1 and 2

FIG 4 is a more detailed block diagram of a portion of the microprocessor shown in FIG 2

FIG 5 is a more detailed block diagram of another portion of the microprocessor shown in FIG 2

FIG. 6 is a block diagram of another portion of the data processing system shown in part in FIG. 3 and incorporating the microprocessor of FIGS. 1–2 and 4–5

FIGS. 7 and 8 are layout diagrams for the data processing system shown in part in FIGS 3 and 6

FIG 9 is a layout diagram of a second embodiment of a microprocessor in accordance with the invention in a data processing system on a single integrated circuit

FIG 10 is a more detailed block diagram of a portion of the data processing system of FIGS 7 and 8

FIG 11 is a timing diagram useful for understanding operation of the system portion shown in FIG 12

FIG 12 is another more detailed block diagram of a further portion of the data processing system of FIGS 7 and 8

FIG. 13 is a more detailed block diagram of a portion of the microprocessor shown in FIG. 2.

FIG. 14 is a more detailed block and schematic diagram of a portion of the system shown in FIGS. 3 and 7–8.

FIG. 15 is a graph useful for understanding operation of the system portion shown in FIG. 14.

FIG. 16 is a more detailed block diagram showing part of the system portion shown in FIG. 4.

FIG. 17 is a more detailed block diagram of a portion of the microprocessor shown in FIG. 2.

FIG. 18 is a more detailed block diagram of part of the microprocessor portion shown in FIG. 17.

FIG. 19 is a set of waveform diagrams useful for understanding operation of the part of the microprocessor portion shown in FIG. 18.

FIG. 20 is a more detailed block diagram showing another part of the system portion shown in FIG. 4.

FIG. 21 is a more detailed block diagram showing another part of the system portion shown in FIG. 4.

FIGS. 22 and 23 are more detailed block diagrams showing another part of the system portion shown in FIG. 4.

DETAILED DESCRIPTION OF THE INVENTION

Overveiw

The microprocessor of this invention is desirably implemented as a 32-bit microprocessor optimized for:

HIGH EXECUTION SPEED, and

LOW SYSTEM COST

In this embodiment, the microprocessor can be thought of as 20 MIPS for 20 dollars. Important distinguishing features of the microprocessor are:

- Uses low-cost commodity DYNAMIC RAMS to run 20 MIPS
- 4 instruction fetch per memory cycle
- On-chip fast page-mode memory management
- Runs fast without external cache
- Requires few interfacing chips
- Crams 32-bit CPU in 44 pin SOJ package

The instruction set is organized so that most operations can be specified with 8-bit instructions. Two positive products of this philosophy are:

- Programs are smaller,
- Programs can execute much faster.

The bottleneck in most computer systems is the memory bus. The bus is used to fetch instructions and fetch and store data. The ability to fetch four instructions in a single memory bus cycle significantly increases the bus availability to handle data.

Turning now to the drawings, more particularly to FIG. 1, there is shown a packaged 32-bit microprocessor **50** in a 44-pin plastic leadless chip carrier, shown approximately 100 times its actual size of about 0.8 inch on a side. The fact that the microprocessor **50** is provided as a 44-pin package represents a substantial departure from typical microprocessor packages, which usually have about 200 input/output (I/O) pins. The microprocessor **50** is rated at 20 million instructions per second (MIPS). Address and data lines **52**, also labelled D0–D31, are shared for addresses and data without speed penalty as a result of the manner in which the microprocessor **50** operates, as will be explained below.

DYNAMIC RAM

In addition to the low cost 44-pin package, another unusual aspect of the high performance microprocessor **50** is that it operates directly with dynamic random access memories (DRAMs), as shown by row address strobe (RAS) and column address strobe (CAS) I/O pins **54**. The other I/O pins for the microprocessor **50** include V_{DD} pins **56**, V_{SS} pins **58**, output enable pin **60**, write pin **62**, clock pin **64** and reset pin **66**.

All high speed computers require high speed and expensive memory to keep up. The highest speed static RAM memories cost as much as ten times as much as slower dynamic RAMs. This microprocessor has been optimized to use low-cost dynamic RAM in high-speed page-mode. Page-mode dynamic RAMs offer static RAM performance without the cost penalty. For example, low-cost 85 nsec. dynamic RAMs access at 25 nsec when operated in fast page-mode. Integrated fast page-mode control on the microprocessor chip simplifies system interfacing and results in a faster system.

Details of the microprocessor **50** are shown in FIG. 2. The microprocessor **50** includes a main central processing unit (CPU) **70** and a separate direct memory access (DMA) CPU **72** in a single integrated circuit making up the microprocessor **50**. The main CPU **70** has a first **16** deep push down stack **74**, which has a top item register **76** and a next item register **78**, respectively connected to provide inputs to an arithmetic logic unit (ALU) **80** by lines **82** and **84**. An output of the ALU **80** is connected to the top item register **76** by line **86**. The output of the top item register at **82** is also connected by line **88** to an internal data bus **90**.

A loop counter **92** is connected to a decrementer **94** by lines **96** and **98**. The loop counter **92** is bidirectionally connected to the internal data bus **90** by line **100**. Stack pointer **102**, return stack pointer **104**, mode register **106** and instruction register **108** are also connected to the internal data bus **90** by lines **110**, **112**, **114** and **116**, respectively. The internal data bus **90** is connected to memory controller **118** and to gate **120**. The gate **120** provides inputs on lines **122**, **124**, and **126** to X register **128**, program counter **130** and Y register **132** of return push down stack **134**. The X register **128**, program counter **130** and Y register **132** provide outputs to internal address bus **136** on lines **138**, **140** and **142**. The internal address bus provides inputs to the memory controller **118** and to an incrementer **144**. The incrementer **144** provides inputs to the X register, program counter and Y register via lines **146**, **122**, **124** and **126**. The DMA CPU **72** provides inputs to the memory controller **118** on line **148**. The memory controller **118** is connected to a RAM (not shown) by address/data bus **150** and control lines **152**.

FIG. 2 shows that the microprocessor **50** has a simple architecture. Prior art RISC microprocessors are substantially more complex in design. For example, the SPARC RISC microprocessor has three times the gates of the microprocessor **50**, and the Intel 8960 RISC microprocessor has 20 times the gates of the microprocessor **50**. The speed of this microprocessor is in substantial part due to this simplicity. The architecture incorporates push down stacks and register write to achieve this simplicity.

The microprocessor **50** incorporates an I/O that has been tuned to make heavy use of resources provided on the integrated circuit chip. On chip latches allow use of the same I/O circuits to handle three different things: column addressing, row addressing and data, with a slight to non-existent speed penalty. This triple bus multiplexing results in fewer buffers to expand, fewer interconnection lines, fewer I/O pins and fewer internal buffers.

The provision of on-chip DRAM control gives a performance equal to that obtained with the use of static RAMs. As a result, memory is provided at ¼ the system cost of static RAM used in most RISC systems.

The microprocessor **50** fetches 4 instructions per memory cycle; the instructions are in an 8-bit format, and this is a 32-bit microprocessor. System speed is therefore 4 times the memory bus bandwidth. This ability enables the microprocessor to break the Von Neumann bottleneck of the speed of getting the next instruction. This mode of operation is possible because of the use of a push down stack and register array. The push down stack allows the use of implied addresses, rather than the prior art technique of explicit addresses for two sources and a destination.

Most instructions execute in 20 nanoseconds in the microprocessor **50**. The microprocessor can therefore execute instructions at 50 peak MIPS without pipeline delays. This is a function of the small number of gates in the microprocessor **50** and the high degree of parallelism in the architecture of the microprocessor.

FIG. 3 shows how column and row addresses are multiplexed on lines D8–D14 of the microprocessor **50** for addressing DRAM **150** from I/O pins **52**. The DRAM **150** is one of eight, but only one DRAM **150** has been shown for clarity. As shown, the lines D11–D18 are respectively connected to row address inputs A0–A8 of the DRAM **150**. Additionally, lines D12–D15 are connected to the data inputs DQ1–DQ4 of the DRAM **150**. The output enable, write and column address strobe pins **54** are respectively connected to the output enable, write and column address strobe inputs of the DRAM **150** by lines **152**. The row address strobe pin **54** is connected through row address strobe decode logic **154** to the row address strobe input of the DRAM **150** by lines **156** and **158**.

D0–D7 pins **52** (FIG. 1) are idle when the microprocessor **50** is outputting multiplexed row and column addresses on D11–D18 pins **52**. The D0–D7 pins **52** can therefore simultaneously be used for I/O when right justified I/O is desired. Simultaneous addressing and I/O can therefore be carried out.

FIG. 4 shows how the microprocessor **50** is able to achieve performance equal to the use of static RAMS with DRAMs through multiple instruction fetch in a single clock cycle and instruction fetch-ahead. Instruction register **108** receives four 8-bit byte instruction words 1–4 on 32-bit internal data bus **90**. The four instruction byte 1–4 locations of the instruction register **108** are connected to multiplexer **170** by busses **172**, **174**, **176** and **178**, respectively. A microprogram counter **180** is connected to the multiplexer **170** by lines **182**. The multiplexer **170** is connected to decoder **184** by bus **186**. The decoder **184** provides internal signals to the rest of the microprocessor **50** on lines **188**.

Most significant bits **190** of each instruction byte 1–4 location are connected to a 4-input decoder **192** by lines **194**. The output of decoder **192** is connected to memory controller **118** by line **196**. Program counter **130** is connected to memory controller **118** by internal address bus **136**, and the instruction register **108** is connected to the memory controller **118** by the internal data bus **90**. Address/data bus **198** and control bus **200** are connected to the DRAMS **150** (FIG. 3).

In operation, when the most significant bits **190** of remaining instructions 1–4 are "1" in a clock cycle of the microprocessor **50**, there are no memory reference instructions in the queue. The output of decoder **192** on line **196** requests an instruction fetch ahead by memory controller **118** without interference with other accesses. While the current instructions in instruction register **108** are executing, the memory controller **118** obtains the address of the next set of four instructions from program counter **130** and obtains that set of instructions. By the time the current set of instructions has completed execution, the next set of instructions is ready for loading into the instruction register.

Details of the DMA CPU **72** are provided in FIG. 5. Internal data bus **90** is connected to memory controller **118** and to DMA instruction register **210**. The DMA instruction register **210** is connected to DMA program counter **212** by bus **214**, to transfer size counter **216** by bus **218** and to timed transfer interval counter **220** by bus **222**. The DMA instruction register **210** is also connected to DMA I/O and RAM address register **224** by line **226**. The DMA I/O and RAM address register **224** is connected to the memory controller **118** by memory cycle request line **228** and bus **230**. The DMA program counter **212** is connected to the internal address bus **136** by bus **232**. The transfer size counter **216** is connected to a DMA instruction done decrementer **234** by lines **236** and **238**. The decrementer **234** receives a control input on memory cycle acknowledge line **240**. When transfer size counter **216** has completed its count, it provides a control signal to DMA program counter **212** on line **242**. Timed transfer interval counter **220** is connected to decrementer **244** by lines **246** and **248**. The decrementer **244** receives a control input from a microprocessor system clock on line **250**.

The DMA CPU **72** controls itself and has the ability to fetch and execute instructions. It operates as a co-processor to the main CPU **70** (FIG. 2) for time specific processing.

FIG. 6 shows how the microprocessor **50** is connected to an electrically programmable read only memory (EPROM) **260** by reconfiguring the data lines **52** so that some of the data lines **52** are input lines and some of them are output lines. Data lines **52** D0–D7 provide data to and from corresponding data terminals **262** of the EPROM **260**. Data lines **52** D9–D18 provide addresses to address terminals **264** of the EPROM **260**. Data lines **52** D19–D31 provide inputs from the microprocessor **50** to memory and I/O decode logic **266**. RAS 0/1 control line **268** provides a control signal for determining whether the memory and I/O decode logic provides a DRAM RAS output on line **270** or a column enable output for the EPROM **260** on line **272**. Column address strobe terminal **60** of the microprocessor **50** provides an output enable signal on line **274** to the corresponding terminal **276** of the EPROM **260**.

FIGS. 7 and 8 show the front and back of a one card data processing system **280** incorporating the microprocessor **50**, MSM514258-10 type DRAMs **150** totalling 2 megabytes, a Motorola 50 MegaHertz crystal oscillator clock **282**, I/O circuits **284** and a 27256 type EPROM **260**. The I/O circuits **284** include a 74HC04 type high speed hex inverter circuit **286**, an IDT39C828 type 10-bit inverting buffer circuit **288**, an IDT39C822 type 10-bit inverting register circuit **290**, and two IDT39C823 type 9-bit non-inverting register circuits **292**. The card **280** is completed with a MAX12V type DC-DC converter circuit **294**, 34-pin dual AMP type headers **296**, a coaxial female power connector **298**, and a 3-pin AMP right angle header **300**. The card **280** is a low cost imbeddable product that can be incorporated in larger systems or used as an internal development tool.

The microprocessor **50** is a very high performance (50 MHz) RISC influenced 32-bit CPU designed to work closely with dynamic RAM. Clock for clock, the microprocessor **50** approaches the theoretical performance limits possible with a single CPU configuration. Eventually, the microprocessor **50** and any other processor is limited by the bus bandwidth and the number of bus paths. The critical conduit is between the CPU and memory.

One solution to the bus bandwidth/bus path problem is to integrate a CPU directly onto the memory chips, giving every memory a direct bus the CPU. FIG. 9 shows another microprocessor **310** that is provided integrally with 1 mega-

bit of DRAM 311 in a single integrated circuit 312. Until the present invention, this solution has not been practical, because most high performance CPUs require from 500,000 to 1,000,000 transistors and enormous die sizes just by themselves. The microprocessor 310 is equivalent to the microprocessor 50 in FIGS. 1–8. The microprocessors 50 and 310 are the most transistor efficient high performance CPUs in existence, requiring fewer than 50,000 transistors for dual processors 70 and 72 (FIG. 2) or 314 and 316 (less memory). The very high speed of the microprocessors 50 and 310 is to a certain extent a function of the small number of active devices. In essence, the less silicon gets in the way the faster the electrons can get where they are going.

The microprocessor 310 is therefore the only CPU suitable for integration on the memory chip die 312. Some simple modifications to the basic microprocessor 50 to take advantage of the proximity to the DRAM array 311 can also increase the microprocessor 50 clock speed by 50 percent, and probably more.

The microprocessor 310 core on board the DRAM die 312 provides most of the speed and functionality required for a large group of applications from automotive to peripheral control. However, the integrated CPU 310/DRAM 311 concept has the potential to redefine significantly the way multiprocessor solutions can solve a spectrum of very compute intensive problems. The CPU 310/DRAM 311 combination eliminates the Von Neumann bottleneck by distributing it across numerous CPU/DRAM chips 312. The microprocessor 310 is a particularly good core for multiprocessing, since it was designed with the SDI targeting array in mind, and provisions were made for efficient interprocessor communications.

Traditional multiprocessor implementations have been very expensive in addition to being unable to exploit fully the available CPU horsepower. Multiprocessor systems have typically been built up from numerous board level or box level computers. The result is usually an immense amount of hardware with corresponding wiring, power consumption and communications problems. By the time the systems are interconnected, as much as 50 percent of the bus speed has been utilized just getting through the interfaces.

In addition, multiprocessor system software has been scarce. A multiprocessor system can easily be crippled by an inadequate load-sharing algorithm in the system software, which allows one CPU to do a great deal of work and the others to be idle. Great strides have been made recently in systems software, and even UNIX V.4 may be enhanced to support multiprocessing. Several commercial products from such manufacturers as DUAL Systems and UNISOFT do a credible job on 68030 type microprocessor systems now.

The microprocessor 310 architecture eliminates most of the interface friction, since up to 64 CPU 310/RAM 311 processors should be able to intercommunicate without buffers or latches. Each chip 312 has about 40 MIPS raw speed, because placing the DRAM 311 next to the CPU 310 allows the microprocessor 310 instruction cycle to be cut in half, compared to the microprocessor 50. A 64 chip array of these chips 312 is more powerful than any other existing computer. Such an array fits on a 3×5 card, cost less than a FAX machine, and draw about the same power as a small television.

Dramatic changes in price/performance always reshape existing applications and almost always create new ones. The introduction of microprocessors in the mid 1970s created video games, personal computers, automotive computers, electronically controlled appliances, and low cost computer peripherals.

The integrated circuit 312 will find applications in all of the above areas, plus create some new ones. A common generic parallel processing algorithm handles convolution/Fast Fourier Transform (FFT)/pattern recognition. Interesting product possibilities using the integrated circuit 312 include high speed reading machines, real-time speech recognition, spoken language translation, real-time robot vision, a product to identify people by their faces, and an automotive or aviation collision avoidance system.

A real time processor for enhancing high density television (HDTV) images, or compressing the HDTV information into a smaller bandwidth, would be very feasible. The load sharing in HDTV could be very straightforward. Splitting up the task according to color and frame would require 6, 9 or 12 processors. Practical implementation might require 4 meg RAMs integrated with the microprocessor 310.

The microprocessor 310 has the following specifications:

CONTROL LINES
4—POWER/GROUND
1—CLOCK
32—DATA I/O
4—SYSTEM CONTROL
- EXTERNAL MEMORY FETCH
- EXTERNAL MEMORY FETCH AUTOINCREMENT X
- EXTERNAL MEMORY FETCH AUTOINCREMENT Y
- EXTERNAL MEMORY WRITE
- EXTERNAL MEMORY WRITE AUTOINCREMENT X
- EXTERNAL MEMORY WRITE AUTOINCREMENT Y
- EXTERNAL PROM FETCH
- LOAD ALL X REGISTERS
- LOAD ALL Y REGISTERS
- LOAD ALL PC REGISTERS
- EXCHANGE X AND Y
- INSTRUCTION FETCH
- ADD TO PC
- ADD TO X
- WRITE MAPPING REGISTER
- READ MAPPING REGISTER

REGISTER CONFIGURATION
MICROPROCESSOR 310 CPU 316 CORE
COLUMN LATCH1 (1024 BITS) 32×32 MUX
STACK POINTER (16 BITS)
COLUMN LATCH2 (1024 BITS) 32×32 MUX
RSTACK POINTER (16 BITS)
PROGRAM COUNTER 32 BITS
X0 REGISTER 32 BITS (ACTIVATED ONLY FOR ON-CHIP ACCESSES)
Y0 REGISTER 32 BITS (ACTIVATED ONLY FOR ON-CHIP ACCESSES)
LOOP COUNTER 32 BITS
DMA CPU 314 CORE
DMA PROGRAM COUNTER 24 BITS
INSTRUCTION REGISTER 32 BITS
I/O & RAM ADDRESS REGISTER 32 BITS
TRANSFER SIZE COUNTER 12 BITS
INTERVAL COUNTER 12 BITS

To offer memory expansion for the basic chip 312, an intelligent DRAM can be produced. This chip will be optimized for high speed operation with the integrated circuit 312 by having three on-chip address registers: Program Counter, X Register and Y register. As a result, to access the intelligent DRAM, no address is required, and a total access cycle could be as short as 10 nsec. Each

expansion DRAM would maintain its own copy of the three registers and would be identified by a code specifying its memory address Incrementing and adding to the three registers will actually take place on the memory chips. A maximum of 64 intelligent DRAM peripherals would allow a large system to be created without sacrificing speed by introducing multiplexers or buffers

There are certain differences between the microprocessor 310 and the microprocessor 50 that arise from providing the microprocessor 310 on the same die 312 with the DRAM 311. Integrating the DRAM 311 allows architectural changes in the microprocessor 310 logic to take advantage of existing on-chip DRAM 311 circuitry Row and column design is inherent in memory architecture. The DRAMs 311 access random bits in a memory array by first selecting a row of 1024 bits, storing them into a column latch, and then selecting one of the bits as the data to be read or written

The time required to access the data is split between the row access and the column access Selecting data already stored in a column latch is faster than selecting a random bit by at least a factor of six. The microprocessor 310 takes advantage of this high speed by creating a number of column latches and using them as caches and shift registers Selecting a new row of information may be thought of as performing a 1024-bit read or write with the resulting immense bus bandwidth

1 The microprocessor **50** treats its 32-bit instruction register **108** (see FIGS **2** and **4**) as a cache for four 8-bit instructions Since the DRAM **311** maintains a 1024-bit latch for the column bits, the microprocessor **310** treats the column latch as a cache for 128 8-bit instructions Therefore, the next instruction will almost always be already present in the cache Long loops within the cache are also possible and more useful than the 4 instruction loops in the microprocessor **50**.

2 The microprocessor **50** uses two 16×32-bit deep register arrays **74** and **134** (FIG **2**) for the parameter stack and the return stack The microprocessor **310** creates two other 1024-bit column latches to provide the equivalent of two 32×32-bit arrays which can be accessed twice as fast as a register array

3 The microprocessor **50** has a DMA capability which can be used for I/O to a video shift register The microprocessor **310** uses yet another 1024-bit column latch as a long video shift register to drive a CRT display directly For color displays, three on-chip shift registers could also be used. These shift registers can transfer pixels at a maximum of 100 MHz.

4 The microprocessor **50** accesses memory via an external 32-bit bus. Most of the memory **311** for the microprocessor **310** is on the same die **312**. External access to more memory is made using an 8-bit bus The result is a smaller die, smaller package and lower power consumption than the microprocessor **50**.

5 The microprocessor **50** consumes about a third of its operating power charging and discharging the I/O pins and associated capacitances The DRAMs **150** (FIG. **8**) connected to the microprocessor **50** dissipate most of their power in the I/O drivers A microprocessor **310** system will consume about one-tenth the power of a microprocessor **50** system, since having the DRAM **311** next to the processor **310** eliminates most of the external capacitances to be charged and discharged

6 Multiprocessing means splitting a computing task between numerous processors in order to speed up the solution. The popularity of multiprocessing is limited by the expense of current individual processors as well as the limited interprocessor communications ability The microprocessor **310** is an excellent multiprocessor candidate, since the chip **312** is a monolithic computer complete with memory, rendering it low-cost and physically compact

The shift registers implemented with the microprocessor **310** to perform video output can also be configured as interprocessor communication links The INMOS transputer attempted a similar strategy, but at much lower speed and without the performance benefits inherent in the microprocessor **310** column latch architecture Serial I O is a prerequisite for many multiprocessor topologies because of the many neighbor processors which communicate A cube has 6 neighbors Each neighbor communicates using these lines:

DATA IN
CLOCK IN
READY FOR DATA
DATA OUT
DATA READY?
CLOCK OUT

A special start up sequence is used to initialize the on-chip DRAM **311** in each of the processors

The microprocessor **310** column latch architecture allows neighbor processors to deliver information directly to internal registers or even instruction caches of other chips **312** This technique is not used with existing processors, because it only improves performance in a tightly coupled DRAM system

7 The microprocessor **50** architecture offers two types of looping structures: LOOP-IF-DONE and MICRO-LOOP The former takes an 8-bit to 24-bit operand to describe the entry point to the loop address The latter performs a loop entirely within the 4 instruction queue and the loop entry point is implied as the first instruction in the queue Loops entirely within the queue run without external instruction fetches and execute up to three times as fast as the long loop construct The microprocessor **310** retains both constructs with a few differences The microprocessor **310** microloop functions in the same fashion as the microprocessor **50** operation, except the queue is 1024-bits or 128 8-bit instructions long The microprocessor **310** microloop can therefore contain jumps, branches, calls and immediate operations not possible in the 4 8-bit instruction microprocessor **50** queue

Microloops in the microprocessor **50** can only perform simple block move and compare functions The larger microprocessor **310** queue allows entire digital signal processing or floating point algorithms to loop at high speed in the queue

The microprocessor **50** offers four instructions to redirect execution:

CALL
BRANCH
BRANCH-IF-ZERO
LOOP-IF-NOT-DONE

These instructions take a variable length address operand 8, 16 or 24 bits long. The microprocessor **50** next address logic treats the three operands similarly by adding or subtracting them to the current program counter For the microprocessor **310**, the 16 and 24-bit operands function in the same manner as the 16 and 24-bit operands in the microprocessor **50** The 8-bit class operands are reserved to operate entirely within the instruction queue Next address decisions can therefore be made quickly, because only 10 bits of addresses are affected, rather than 32 There is no carry or borrow generated past the 10 bits

8 The microprocessor **310** CPU **316** resides on an already crowded DRAM die **312** To keep chip size as small as

possible, the DMA processor 72 of the microprocessor 50 has been replaced with a more traditional DMA controller 314. DMA is used with the microprocessor 310 to perform the following functions:

Video output to a CRT

Multiprocessor serial communications

8-bit parallel I O

The DMA controller 314 can maintain both serial and parallel transfers simultaneously The following DMA sources and destinations are supported by the microprocessor 310

DESCRIPTION	I/O	LINES
1 Video shift register	OUTPUT	1 to 3
2 Multiprocessor serial	BOTH	6 lines, channel
3 8-bit parallel	BOTH	8 data, 4 control

The three sources use separate 1024-bit buffers and separate I/O pins Therefore, all three may be active simultaneously without interference

The microprocessor 310 can be implemented with either a single multiprocessor serial buffer or separate receive and sending buffers for each channel, allowing simultaneous bidirectional communications with six neighbors simultaneously

FIGS 10 and 11 provide details of the PROM DMA used in the microprocessor 50 The microprocessor 50 executes faster than all but the fastest PROMs PROMS are used in a microprocessor 50 system to store program segments and perhaps entire programs The microprocessor 50 provides a feature on power-up to allow programs to be loaded from low-cost slow speed PROMs into high speed DRAM for execution The logic which performs this function is part of the DMA memory controller 118 The operation is similar to DMA, but not identical, since four 8-bit bytes must be assembled on the microprocessor 50 chip then written to the DRAM 150

The microprocessor 50 directly interfaces to DRAM 150 over a triple multiplexed data and address bus 350 which carries RAS addresses, CAS addresses and data The EPROM 260, on the other hand, is read with non-multiplexed busses The microprocessor 50 therefore has a special mode which unmultiplexes the data and address lines to read 8 bits of EPROM data Four 8-bit bytes are read in this fashion The multiplexed bus 350 is turned back on and the data is written to the DRAM 150

When the microprocessor 50 detects a RESET condition, the processor stops the main CPU 70 and forces a mode 0 (PROM LOAD) instruction into the DMA CPU 72 instruction register The DMA instruction directs the memory controller to read the EPROM 260 data at 8 times the normal access time for memory Assuming a 50 MHz microprocessor 50, this means an access time of 320 nsec The instruction also indicates:

The selection address of the EPROM 260 to be loaded.

The number of 32-bit words to transfer.

The DRAM 150 address to transfer into

The sequence of activities to transfer one 32-bit word from EPROM 260 to DRAM 150 are:

1 RAS goes low at 352, latching the EPROM 260 select information from the high order address bits The EPROM 260 is selected

2 Twelve address bits (consisting of what is normally DRAM CAS addresses plus two byte select bits are placed on the bus 350 going to the EPROM 260 address pins These signals will remain on the lines until the data from the EPROM 260 has been read into the microprocessor 50 For the first byte, the byte select bits will be binary 00.

3 CAS goes low at 354 enabling the EPROM 260 data onto the lower 8 bits of the external address/data bus 350 NOTE: It is important to recognize that, during this part of the cycle, the lower 8 bits of the external data/address bus are functioning as inputs but the rest of the bus is still acting as outputs

4 The microprocessor 50 latches these eight least significant bits internally and shifts them 8 bits left to shift them to the next significant byte position

5 Steps 2, 3 and 4 are repeated with byte address 01

6 Steps 2, 3 and 4 are repeated with byte address 10

7 Steps 2, 3 and 4 are repeated with byte address 11

8 CAS goes high at 356, taking the EPROM 260 off the data bus

9 RAS goes high at 358, indicating the end of the EPROM 260 access

10. RAS goes low at 360, latching the DRAM select information from the high order address bits At the same time, the RAS address bits are latched into the DRAM 150 The DRAM 150 is selected

11 CAS goes low at 362 latching the DRAM 150 CAS addresses

12. The microprocessor 50 places the previously latched EPROM 260 32-bit data onto the external address/data bus 350 W goes low at 364, writing the 32 bits into the DRAM 150

13 W goes high at 366 CAS goes high at 368 The process continues with the next word

FIG. 12 shows details of the microprocessor 50 memory controller 118 In operation, bus requests stay present until they are serviced CPU 70 requests are prioritized at 370 in the order of: 1, Parameter Stack; 2, Return Stack; 3, Data Fetch; 4, Instruction Fetch The resulting CPU request signal and a DMA request signal are supplied as bus requests to bus control 372 which provides a bus grant signal at 374 Internal address bus 136 and a DMA counter 376 provide inputs to a multiplexer 378 Either a row address or a column address are provided as an output to multiplexed address bus 380 as an output from the multiplexer 378. The multiplexed address bus 380 and the internal data bus 90 provide address and data inputs, respectively, to multiplexer 382 Shift register 384 supplies row address strobe (RAS) 1 and 2 control signals to multiplexer 386 and column address strobe (CAS) 1 and 2 control signals to multiplexer 388 on lines 390 and 392. The shift register 384 also supplies output enable (OE) and write (W) signals on lines 394 and 396 and a control signal on line 398 to multiplexer 382 The shift register 384 receives a RUN signal on line 400 to generate a memory cycle and supplies a MEMORY READY signal on line 402 when an access is complete

STACK/REGISTER ARCHITECTURE

Most microprocessors use on-chip registers for temporary storage of variables The on-chip registers access data faster than off-chip RAM A few microprocessors use an on-chip push down stack for temporary storage

A stack has the advantage of faster operation compared to on-chip registers by avoiding the necessity to select source and destination registers (A math or logic operation always uses the top two stack items as source and the top of stack as destination) The stack's disadvantage is that it makes some operations clumsy Some compiler activities in particular require on-chip registers for efficiency

As shown in FIG. 13, the microprocessor **50** provides both on-chip registers **134** and a stack **74** and reaps the benefits of both.

BENEFITS:

1. Stack math and logic is twice as fast as those available on an equivalent register only machine. Most programmers and optimizing compilers can take advantage of this feature.
2. Sixteen registers are available for on-chip storage of local variables which can transfer to the stack for computation. The accessing of variables is three to four times as fast as available on a strictly stack machine.

The combined stack **74**/register **134** architecture has not been used previously due to inadequate understanding by computer designers of optimizing compilers and the mix of transfer versus math/logic instructions.

ADAPTIVE MEMORY CONTROLLER

A microprocessor must be designed to work with small or large memory configurations. As more memory loads are added to the data, address, and control lines, the switching speed of the signals slows down. The microprocessor **50** multiplexes the address/data bus three ways, so timing between the phases is critical. A traditional approach to the problem allocates a wide margin of time between bus phases so that systems will work with small or large numbers of memory chips connected. A speed compromise of as much as 50% is required.

As shown in FIG. 14, the microprocessor **50** uses a feedback technique to allow the processor to adjust memory bus timing to be fast with small loads and slower with large ones. The OUTPUT ENABLE (OE) line **152** from the microprocessor **50** is connected to all memories **150** on the circuit board. The loading on the output enable line **152** to the microprocessor **50** is directly related to the number of memories **150** connected. By monitoring how rapidly OE **152** goes high after a read, the microprocessor **50** is able to determine when the data hold time has been satisfied and place the next address on the bus.

The level of the OE line **152** is monitored by CMOS input buffer **410** which generates an internal READY signal on line **412** to the microprocessor's memory controller. Curves **414** and **416** of the FIG. 15 graph show the difference in rise time likely to be encountered from a lightly to heavily loaded memory system. When the OE line **152** has reached a predetermined level to generate the READY signal, driver **418** generates an OUTPUT ENABLE signal on OE line **152**.

SKIP WITHIN THE INSTRUCTION CACHE

The microprocessor **50** fetches four 8-bit instructions each memory cycle and stores them in a 32-bit instruction register **108**, as shown in FIG. 16. A class of "test and skip" instructions can very rapidly execute a very fast jump operation within the four instruction cache.

SKIP CONDITIONS:

Always

ACC non-zero

ACC negative

Carry flag equal logic one

Never

ACC equal zero

ACC positive

Carry flag equal logic zero

The SKIP instruction can be located in any of the four byte positions **420** in the 32-bit instruction register **108**. If the test is successful, SKIP will jump over the remaining one, two, or three 8-bit instructions in the instruction register **108** and cause the next four-instruction group to be loaded into the register **108**. As shown, the SKIP operation is implemented by resetting the 2-bit microinstruction counter **180** to zero on line **422** and simultaneously latching the next instruction group into the register **108**. Any instructions following the SKIP in the instruction register are overwritten by the new instructions and not executed.

The advantage of SKIP is that optimizing compilers and smart programmers can often use it in place of the longer conditional JUMP instruction. SKIP also makes possible microloops which exit when the loop counts down or when the SKIP jumps to the next instruction group. The result is very fast code.

Other machines (such as the PDP-8 and Data General NOVA) provide the ability to skip a single instruction. The microprocessor **50** provides the ability to skip up to three instructions.

MICROLOOP IN THE INSTRUCTION CACHE

The microprocessor **50** provides the MICROLOOP instruction to execute repetitively from one to three instructions residing in the instruction register **108**. The microloop instruction works in conjunction with the LOOP COUNTER **92** (FIG. 2) connected to the internal data bus **90**. To execute a microloop, the program stores a count in LOOP COUNTER **92**. MICROLOOP may be placed in the first, second, third, or last byte **420** of the instruction register **108**. If placed in the first position, execution will just create a delay equal to the number stored in LOOP COUNTER **92** times the machine cycle. If placed in the second, third, or last byte **420**, when the microloop instruction is executed, it will test the LOOP COUNT for zero. If zero, execution will continue with the next instruction. If not zero, the LOOP COUNTER **92** is decremented and the 2-bit microinstruction counter is cleared, causing the preceding instructions in the instruction register to be executed again.

Microloop is useful for block move and search operations. By executing a block move completely out of the instruction register **108**, the speed of the move is doubled, since all memory cycles are used by the move rather than being shared with instruction fetching. Such a hardware implementation of microloops is much faster than conventional software implementation of a comparable function.

OPTIMAL CPU CLOCK SCHEME

The designer of a high speed microprocessor must produce a product which operate over wide temperature ranges, wide voltage swings, and wide variations in semiconductor processing. Temperature, voltage, and process all affect transistor propagation delays. Traditional CPU designs are done so that with the worse case of the three parameters, the circuit will function at the rated clock speed. The result are designs that must be clocked a factor of two slower than their maximum theoretical performance, so they will operate properly in worse case conditions.

The microprocessor **50** uses the technique shown in FIGS. 17–19 to generate the system clock and its required phases. Clock circuit **430** is the familiar "ring oscillator" used to test process performance. The clock is fabricated on the same silicon chip as the rest of the microprocessor **50**.

The ring oscillator frequency is determined by the parameters of temperature, voltage, and process. At room temperature, the frequency will be in the neighborhood of 100 MHZ. At 70 degrees Centigrade, the speed will be 50 MHZ. The ring oscillator **430** is useful as a system clock, with its stages **431** producing phase 0-phase 3 outputs **433** shown in FIG. 19, because its performance tracks the parameters which similarly affect all other transistors on the same silicon die. By deriving system timing from the ring

oscillator **430**, CPU **70** will always execute at the maximum frequency possible but never too fast For example, if the processing of a particular die is not good resulting in slow transistors, the latches and gates on the microprocessor **50** will operate slower than normal Since the microprocessor **50** ring oscillator clock **430** is made from the same transistors on the same die as the latches and gates, it too will operate slower (oscillating at a lower frequency), providing compensation which allows the rest of the chip's logic to operate properly.

ASYNCHRONOUS/SYNCHRONOUS CPU

Most microprocessors derive all system timing from a single clock The disadvantage is that different parts of the system can slow all operations The microprocessor **50** provides a dual-clock scheme as shown in FIG **17**, with the CPU **70** operating a synchronously to I/O interface **432** forming part of memory controller **118** (FIG 2) and the I/O interface **432** operating synchronously with the external world of memory and I/O devices The CPU **70** executes at the fastest speed possible using the adaptive ring counter clock **430** Speed may vary by a factor of four depending upon temperature, voltage, and process The external world must be synchronized to the microprocessor **50** for operations such as video display updating and disc drive reading and writing. This synchronization is performed by the I/O interface **432**, speed of which is controlled by a conventional crystal clock **434** The interface **432** processes requests for memory accesses from the microprocessor **50** and acknowledges the presence of I/O data The microprocessor **50** fetches up to four instructions in a single memory cycle and can perform much useful work before requiring another memory access By decoupling the variable speed of the CPU **70** from the fixed speed of the I/O interface **432** optimum performance can be achieved by each Recoupling between the CPU **70** and the interface **432** is accomplished with handshake signals on lines **436**, with data/addresses passing on bus **90, 136**.

ASYNCHRONOUS/SYNCHRONOUS CPU IMBEDDED ON A DRAM CHIP

System performance is enhanced even more when the DRAM **311** and CPU **314** (FIG 9) are located on the same die. The proximity of the transistors means that DRAM **311** and CPU **314** parameters will closely follow each other. At room temperature, not only would the CPU **314** execute at 100 MHZ, but the DRAM **311** would access fast enough to keep up The synchronization performed by the I/O interface **432** would be for DMA and reading and writing I/O ports In some systems (such as calculators) no I/O synchronization at all would be required and the I/O clock would be tied to the ring counter clock

VARIABLE WIDTH OPERANDS

Many microprocessors provide variable width operands The microprocessor **50** handles operands of 8, 16, or 24 bits using the same op-code. FIG **20** shows the 32-bit instruction register **108** and the 2-bit microinstruction register **180** which selects the 8-bit instruction. Two classes of microprocessor **50** instructions can be greater than 8-bits, JUMP class and IMMEDIATE A JUMP or IMMEDIATE op-code is 8-bits but the operand can be 8 16 or 24 bits long. This magic is possible because operands must be right justified in the instruction register This means that the least significant bit of the operand is always located in the least significant bit of the instruction register The microinstruction counter **180** selects which 8-bit instruction to execute If a JUMP or IMMEDIATE instruction is decoded, the state of the 2-bit microinstruction counter selects the required 8, 16, or 24 bit operand onto the address or data bus The unselected 8-bit bytes are loaded with zeros by operation of decoder **440** and gates **442** The advantage of this technique is the saving of a number of op-codes required to specify the different operand sizes in other microprocessors.

TRIPLE STACK CACHE

Computer performance is directly related to the system memory bandwidth The faster the memories, the faster the computer Fast memories are expensive, so techniques have been developed to move a small amount of high-speed memory around to the memory addresses where it is needed A large amount of slow memory is constantly updated by the fast memory, giving the appearance of a large fast memory array. A common implementation of the technique is known as a high-speed memory cache The cache may be thought of as fast acting shock absorber smoothing out the bumps in memory access. When more memory is required than the shock can absorb, it bottoms out and slow speed memory is accessed. Most memory operations can be handled by the shock absorber itself

The microprocessor **50** architecture has the ALU **80** (FIG 2) directly coupled to the top two stack locations **76** and **78** The access time of the stack **74** therefore directly affects the execution speed of the processor The microprocessor **50** stack architecture is particularly suitable to a triple cache technique shown in FIG **21** which offers the appearance of a large stack memory operating at the speed of on-chip latches **450** Latches **450** are the fastest form of memory device built on the chip, delivering data in as little as 3 nsec However latches **450** require large numbers of transistors to construct On-chip RAM **452** requires fewer transistors than latches but is slower by a factor of five (15 nsec access) Off-chip RAM **150** is the slowest storage of all The microprocessor **50** organizes the stack memory hierarchy as three interconnected stacks **450, 452** and **454**. The latch stack **450** is the fastest and most frequently used The on-chip RAM stack **452** is next The off-chip RAM stack **454** is slowest. The stack modulation determines the effective access time of the stack If a group of stack operations never push or pull more than four consecutive items on the stack, operations will be entirely performed in the 3 nsec latch stack When the four latches **456** are filled, the data in the bottom of the latch stack **450** is written to the top of the on-chip RAM stack **452** When the sixteen locations **458** in the on-chip RAM stack **452** are filled, the data in the bottom of the on-chip RAM stack **452** is written to the top of the off-chip RAM stack **454** When popping data off a full stack **450**, four pops will be performed before stack empty line **460** from the latch stack pointer **462** transfers data from the on-chip RAM stack **452** By waiting for the latch stack **450** to empty before performing the slower on-chip RAM access, the high effective speed of the latches **456** are made available to the processor The same approach is employed with the on-chip RAM stack **452** and the off-chip RAM stack **454**

POLYNOMIAL GENERATION INSTRUCTION

Polynomials are useful for error correction, encryption, data compression and fractal generation A polynomial is generated by a sequence of shift and exclusive OR operations Special chips are provided for this purpose in the prior art

The microprocessor **50** is able to generate polynomials at high speed without external hardware by slightly modifying how the ALU **80** works. As shown in FIG **21**, a polynomial is generated by loading the "order" (also known as the feedback terms) into C Register **470** The value thirty one (resulting in 32 iterations) is loaded into DOWN COUNTER **472**. A register **474** is loaded with zero B register **476** is loaded with the starting polynomial value When the POLY

instruction executes, C register **470** is exclusively ORed with A register **474** if the least significant bit of B register **476** is a one. Otherwise, the contents of the A register **474** passes through the ALU **80** unaltered. The combination of A and B is then shifted right (divided by 2) with shifters **478** and **480**. The operation automatically repeats the specified number of iterations, and the resulting polynomial is left in A register **474**.

FAST MULTIPLY

Most microprocessors offer a 16×16 or 32×32 bit multiply instruction. Multiply when performed sequentially takes one shift/add per bit, or 32 cycles for 32 bit data. The microprocessor **50** provides a high speed multiply which allows multiplication by small numbers using only a small number of cycles. FIG. **23** shows the logic used to implement the high speed algorithm. To perform a multiply, the size of the multiplier less one is placed in the DOWN COUNTER **472**. For a four bit multiplier, the number three would be stored in the DOWN COUNTER **472**. Zero is loaded into the A register **474**. The multiplier is written bit reversed into the B Register **476**. For example, a bit reversed five (binary 0101) would be written into B as 1010. The multiplicand is written into the C register **470**. Executing the FAST MULT instruction will leave the result in the A Register **474**, when the count has been completed. The fast multiply instruction is important because many applications scale one number by a much smaller number. The difference in speed between multiplying a 32×32 bit and a 32×4 bit is a factor of 8. If the least significant bit of the multiplier is a "ONE", the contents of the A register **474** and the C register **470** are added. If the least significant bit of the multiplier is a "ZERO", the contents of the A register are passed through the ALU **80** unaltered. The output of the ALU **80** is shifted left by shifter **482** in each iteration. The contents of the B register **476** are shifted right by the shifter **480** in each iteration.

INSTRUCTION EXECUTION PHILOSOPHY

The microprocessor **50** uses high speed D latches in most of the speed critical areas. Slower on-chip RAM is used as secondary storage.

The microprocessor **50** philosophy of instruction execution is to create a hierarchy of speed as follows:

Logic and D latch transfers	1 cycle	20 nsec
Math	2 cycles	40 nsec
Fetch/store on-chip RAM	2 cycles	40 nsec
Fetch/store in current RAS page	4 cycles	80 nsec
Fetch/store with RAS cycle	11 cycles	220 nsec

With a 50 MHZ clock, many operations can be performed in 20 nsec. and almost everything else in 40 nsec.

To maximize speed, certain techniques in processor design have been used. They include:

Eliminating arithmetic operations on addresses,

Fetching up to four instructions per memory cycle,

Pipelineless instruction decoding

Generating results before they are needed,

Use of three level stack caching

PIPELINE PHILOSOPHY

Computer instructions are usually broken down into sequential pieces, for example: fetch, decode, register read, execute, and store. Each piece will require a single machine cycle. In most Reduced Instruction Set Computer (RISC) chips, instruction require from three to six cycles.

RISC instructions are very parallel. For example, each of 70 different instructions in the SPARC (SUN Computer's RISC chip) has five cycles. Using a technique called pipelining, the different phases of consecutive instructions can be overlapped.

To understand pipelining, think of building five residential homes. Each home will require in sequence, a foundation, framing, plumbing and wiring, roofing, and interior finish. Assume that each activity takes one week. To build one house will take five weeks.

But what if you want to build an entire subdivision? You have only one of each work crew, but when the foundation men finish on the first house, you immediately start them on the second one, and so on. At the end of five weeks, the first home is complete, but you also have five foundations. If you have kept the framing, plumbing, roofing, and interior guys all busy, from five weeks on, a new house will be completed each week.

This is the way a RISC chip like SPARC appears to execute an instruction in a single machine cycle. In reality, a RISC chip is executing one fifth of five instructions each machine cycle. And if five instructions stay in sequence, an instruction will be completed each machine cycle.

The problems with a pipeline are keeping the pipe full with instructions. Each time an out of sequence instruction such as a BRANCH or CALL occurs, the pipe must be refilled with the next sequence. The resulting dead time to refill the pipeline can become substantial when many IF/THEN/ELSE statements or subroutines are encountered.

THE PIPELINE APPROACH

The microprocessor **50** has no pipeline as such. The approach of this microprocessor to speed is to overlap instruction fetching with execution of the previously fetched instruction(s). Beyond that, over half the instructions (the most common ones) execute entirely in a single machine cycle of 20 nsec. This is possible because:

1. Instruction decoding resolves in 2.5 nsec.
2. Incremented/decremented and some math values are calculated before they are needed, requiring only a latching signal to execute.
3. Slower memory is hidden from high speed operations by high-speed D latches which access in 4 nsec.

The disadvantage for this microprocessor is a more complex chip design process. The advantage for the chip user is faster ultimate throughput since pipeline stalls cannot exist. Pipeline synchronization with availability flag bits and other such pipeline handling is not required by this microprocessor.

For example, in some RISC machines an instruction which tests a status flag may have to wait for up to four cycles for the flag set by the previous instruction to be available to be tested. Hardware and software debugging is also somewhat easier because the user doesn't have to visualize five instructions simultaneously in the pipe.

OVERLAPPING INSTRUCTION FETCH/EXECUTE

The slowest procedure the microprocessor **50** performs is to access memory. Memory is accessed when data is read or written. Memory is also read when instructions are fetched. The microprocessor **50** is able to hide fetch of the next instruction behind the execution of the previously fetched instruction(s). The microprocessor **50** fetches instructions in 4-byte instruction groups. An instruction group may contain from one to four instructions. The amount of time required to execute the instruction group ranges from 4 cycles for simple instructions to 64 cycles for a multiply.

When a new instruction group is fetched, the microprocessor instruction decoder looks at the most significant bit of all four of the bytes. The most significant bit of an instruction determines if a memory access is required. For example, CALL, FETCH, and STORE all require a memory access to

execute If all four bytes have nonzero most significant bits, the microprocessor initiates the memory fetch of the next sequential 4-byte instruction group When the last instruction in the group finishes executing, the next 4-byte instruction group is ready and waiting on the data bus needing only to be latched into the instruction register. If the 4-byte instruction group required four or more cycles to execute and the next sequential access was a column address strobe (CAS) cycle, the instruction fetch was completely overlapped with execution.

INTERNAL ARCHITECTURE

The microprocessor **50** architecture consists of the following:

```
PARAMETER STACK      <--->            Y REGISTER
                          ALU         RETURN STACK
                     <--->
<---32 BITS--->                       <---32 BITS--->
    16 DEEP                               16 DEEP
Used for math and logic               Used for subroutine
                                      and interrupt return
                                      addresses as well as
                                      local variables
Push down stack                       Push down stack
Can overflow into                     Can overflow into
off-chip RAM                          off-chip RAM
                                      Can also be accessed
                                      relative to top of
                                      stack
LOOP COUNTER         (32-bits, can decrement by 1)
                     Used by class of test and loop
                     instructions
X REGISTER           (32-bits, can increment or decrement by
                     4). Used to point to RAM locations
PROGRAM COUNTER      (32-bits, increments by 4). Points to
                     4-byte instruction groups in RAM
INSTRUCTION REG      (32-Bits) Holds 4-byte instruction
                     groups while they are being decoded
                     and executed
MODE - A register with mode and status bits
MODE-BITS:
   - Slow down memory accesses by 8 if "1". Run full
speed if "0". (Provided for access to slow EPROM.)
   - Divide the system clock by 1023 if "1" to reduce
     power consumption. Run full speed if "0". (On-chip
     counters slow down if this bit is set.)
     - Enable external interrupt 1
     - Enable external interrupt 2
     - Enable external interrupt 3
     - Enable external interrupt 4
     - Enable external interrupt 5
     - Enable external interrupt 6
     - Enable external interrupt 7
ON-CHIP MEMORY LOCATIONS:
          MODE-BITS
          DMA-POINTER
          DMA-COUNTER
          STACK-POINTER     - Pointer into Parameter Stack.
          STACK-DEPTH       - Depth of on-chip Parameter Stack
          RSTACK-POINTER    - Pointer into Return Stack
          RSTACK-DEPTH      - Depth of on-chip Return Stack
```

*Math and logic operations use the TOP item and NEXT to top Parameter Stack items as the operands. The result is pushed onto the Parameter Stack. *Return addresses from subroutines are placed on the Return Stack. The Y REGISTER is used as a pointer to RAM locations. Since the Y REGISTER is the top item of the Return Stack, nesting of indices is straightforward.

ADDRESSING MODE HIGH POINTS

The data bus is 32-bits wide. All memory fetches and stores are 32-bits. Memory bus addresses are 30 bits. The least significant 2 bits are used to select one-of-four bytes in some addressing modes. The Program Counter, X Register, and Y Register are implemented as D latches with their outputs going to the memory address bus and the bus incrementer/decrementer. Incrementing one of these registers can happen quickly, because the incremented value has already rippled through the inc/dec logic and need only be clocked into the latch. Branches and Calls are made to 32-bit word boundaries.

INSTRUCTION SET

32-BIT INSTRUCTION FORMAT

The thirty two bit instructions are CALL, BRANCH, BRANCH-IF-ZERO, and LOOP-IF-NOT-DONE. These instructions require the calculation of an effective address. In many computers, the effective address is calculated by adding or subtracting an operand with the current Program Counter. This math operation requires from four to seven machine cycles to perform and can definitely bog down machine execution. The microprocessor's strategy is to perform the required math operation at assembly or linking time and do a much simpler "Increment to next page" or "Decrement to previous page" operation at run time. As a result, the microprocessor branches execute in a single cycle.

24-BIT OPERAND FORM:

Byte 1	Byte 2	Byte 3	Byte 4
WWWWWW XX -	YYYYYYYY -	YYYYYYYY -	YYYYYYYY

With a 24-bit operand, the current page is considered to be defined by the most significant 6 bits of the Program Counter.

16-BIT OPERAND FORM: QQQQQQQQ-WWWWWW XX-YYYYYYYY-YYYYYYYY With a 16-bit operand, the current page is considered to be defined by the most significant 14 bits of the Program Counter.

8-BIT OPERAND FORM: QQQQQQQQ-QQQQQQQQ-WWWWWW XX-YYYYYYYY With an 8-bit operand, the current page is considered to be defined by the most significant 22 bits of the Program Counter.

QQQQQQQQ—Any 8-bit instruction

WWWWWW—Instruction op-code

XX—Select how the address bits will be used:

00—Make all high-order bits zero. (Page zero addressing)

01—Increment the high-order bits. (Use next page)

10—Decrement the high-order bits. (Use previous page)

11—Leave the high-order bits unchanged. (Use current page)

YYYYYYYY—The address operand field. This field is always shifted left two bits (to generate a word rather than byte address) and loaded into the Program Counter. The microprocessor instruction decoder figures out the width of the operand field by the location of the instruction op-code in the four bytes.

The compiler or assembler will normally use the shortest operand required to reach the desired address so that the leading bytes can be used to hold other instructions. The effective address is calculated by combining:

The current Program Counter,

The 8, 16, or 24 bit address operand in the instruction

Using one of the four allowed addressing modes.

EXAMPLES OF EFFECTIVE ADDRESS CALCULATION

Example 1

Byte 1	Byte 2	Byte 3	Byte 4
QQQQQQQQ	QQQQQQQQ	00000011	10011000

The "QQQQQQQQs" in Byte 1 and 2 indicate space in the 4-byte memory fetch which could be hold two other

instructions to be executed prior to the CALL instruction Byte 3 indicates a CALL instruction (six zeros) in the current page (indicated by the 11 bits) Byte 4 indicates that the hexadecimal number **98** will be forced into the Program Counter bits **2** through **10**. (Remember, a CALL or BRANCH always goes to a word boundary so the two least significant bits are always set to zero) The effect of this instruction would be to CALL a subroutine at WORD location HEX 98 in the current page. The most significant 22 bits of the Program Counter define the current page and will be unchanged

Example 2

Byte 1	Byte 2	Byte 3	Byte 4
000001 01	00000001	00000000	00000000

If we assume that the Program Counter was HEX 0000 0156 which is binary:

00000000 00000000 00000001 01010110=OLD PROGRAM COUNTER

Byte 1 indicates a BRANCH instruction op code (000001) and "01" indicates select the next page Byte 2.3., and 4 are the address operand These 24-bits will be shifted to the left two places to define a WORD address HEX 0156 shifted left two places is HEX 0558. Since this is a 24-bit operand instruction, the most significant 6 bits of the Program Counter define the current page These six bits will be incremented to select the next page Executing this instruction will cause the Program Counter to be loaded with HEX 0400 0558 which is binary:

00000100 00000000 00000101 01011000=NEW PROGRAM COUNTER

INSTRUCTIONS

CALL-LONG

0000 00XX-YYYYYYYY-YYYYYYYY-YYYYYYYY

Load the Program Counter with the effective WORD address specified Push the current PC contents onto the RETURN STACK.

OTHER EFFECTS: CARRY or modes, no effect May cause Return Stack to force an external memory cycle if on-chip Return Stack is full

BRANCH

0000 01XX-YYYYYYYY-YYYYYYYY-YYYYYYYY

Load the Program Counter with the effective WORD address specified

OTHER EFFECTS: NONE

BRANCH-IF-ZERO

0000 10XX-YYYYYYYY-YYYYYYYY-YYYYYYYY

Test the TOP value on the Parameter Stack If the value is equal to zero, load the Program Counter with the effective WORD address specified If the TOP value is not equal to zero, increment the Program Counter and fetch and execute the next instruction.

OTHER EFFECTS: NONE

LOOP-IF-NOT-DONE

0000 11YY-(XXXX XXXX)-(XXXX XXXX)-(XXXX XXXX)

If the LOOP COUNTER is not zero, load the Program Counter with the effective WORD address specified. If the LOOP COUNTER is zero, decrement the LOOP COUNTER, increment the Program Counter and fetch and execute the next instruction.

OTHER EFFECTS: NONE

8-BIT INSTRUCTIONS PHILOSOPHY

Most of the work in the microprocessor **50** is done by the 8-bit instructions Eight bit instructions are possible with the microprocessor because of the extensive use of implied stack addressing Many 32-bit architectures use 8-bits to specify the operation to perform but use an additional 24-bits to specify two sources and a destination

For math and logic operations, the microprocessor **50** exploits the inherent advantage of a stack by designating the source operand(s) as the top stack item and the next stack item The math or logic operation is performed, the operands are popped from the stack and the result is pushed back on the stack The result is a very efficient utilization of instruction bits as well as registers A comparable situation exists between Hewlett Packard calculators (which use a stack) and Texas Instrument calculators which don't The identical operation on an HP will require one half to one third the keystrokes of the TI.

The availability of 8-bit instructions also allows another architectural innovation the fetching of four instructions in a single 32-bit memory cycle The advantages of fetching multiple instructions are:

Increased execution speed even with slow memories,

Similar performance to the Harvard (separate data and instruction busses) without the expense,

Opportunities to optimize groups of instructions.,

The capability to perform loops within this mini-cache. The microloops inside the four instruction group are effective for searches and block moves

SKIP INSTRUCTIONS

The microprocessor **50** fetches instructions in 32-bit chunks called 4-byte instruction groups These four bytes may contain four 8-bit instructions or some mix of 8-bit and 16 or 24-bit instructions. SKIP instructions in the microprocessor skip any remaining instructions in a 4-byte instruction group and cause a memory fetch to get the next 4-byte instruction group Conditional SKIPs when combined with 3-byte BRANCHES will create conditional BRANCHES SKIPs may also be used in situations when no use can be made of the remaining bytes in a 4-instruction group A SKIP executes in a single cycle, whereas a group of three NOPs would take three cycles

SKIP-ALWAYS—Skip any remaining instructions in this 4-byte instruction group Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group

SKIP-IF-ZERO—If the TOP item of the Parameter Stack is zero, skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the TOP item is not zero execute the next sequential instruction

SKIP-IF-POSITIVE—If the TOP item of the Parameter Stack has a the most significant bit (the sign bit) equal to "0", skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group If the TOP item is not "0" execute the next sequential instruction

SKIP-IF-NO-CARRY—If the CARRY flag from a SHIFT or arithmetic operation is not equal to "1", skip any remaining instructions in the 4-byte instruction group Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group If the CARRY is equal to "1" execute the next sequential instruction

SKIP-NEVER (NOP) execute the next sequential instruction. (Delay one machine cycle).

SKIP-IF-NOT-ZERO—If the TOP item on the Parameter Stack is not equal to "0", skip any remaining instructions

in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the TOP item is equal "0", execute the next sequential instruction.

SKIP-IF-NEGATIVE—If the TOP item on the Parameter Stack has its most significant bit (sign bit) set to "1", skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the TOP item has its most significant bit set to "0", execute the next sequential instruction.

SKIP-IF-CARRY—If the CARRY flag is set to "1" as a result of SHIFT or arithmetic operation, skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the CARRY flag is "0", execute the next sequential instruction.

MICROLOOPS

Microloops are a unique feature of the microprocessor architecture which allows controlled looping within a 4-byte instruction group. A microloop instruction tests the LOOP COUNTER for "0" and may perform an additional test. If the LOOP COUNTER is not "0" and the test is met, instruction execution continues with the first instruction in the 4-byte instruction group, and the LOOP COUNTER is decremented. A microloop instruction will usually be the last byte in a 4-byte instruction group, but it can be any byte. If the LOOP COUNTER is "0" or the test is not met, instruction execution continues with the next instruction. If the microloop is the last byte in the 4-byte instruction group, the most significant 30-bits of the Program Counter are incremented and the next 4-byte instruction group is fetched from memory. On a termination of the loop on LOOP COUNTER equal to "0", the LOOP COUNTER will remain at "0". Microloops allow short iterative work such as moves and searches to be performed without slowing down to fetch instructions from memory.

EXAMPLE

Byte 1	Byte 2
FETCH-VIA-X-AUTO-INCREMENT	STORE-VIA-Y-AUTOINCREMENT
Byte 3	Byte 4
ULOOP-UNTIL-DONE	QQQQQQQQ

This example will perform a block move. To initiate the transfer, X will be loaded with the starting address of the source. Y will be loaded with the starting address of the destination. The LOOP COUNTER will be loaded with the number of 32-bit words to move. The microloop will FETCH and STORE and count down the LOOP COUNTER until it reaches zero. QQQQQQQQ indicates any instruction can follow.

MICROLOOP INSTRUCTIONS

ULOOP-UNTIL-DONE—If the LOOP COUNTER is not "0", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0", continue execution with the next instruction.

ULOOP-IF-ZERO—If the LOOP COUNTER is not "0" and the TOP item on the Parameter Stack is "0", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the TOP item is "1" continue execution with the next instruction.

ULOOP-IF-POSITIVE—If the LOOP COUNTER is not "0" and the most significant bit (sign bit) is "0", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the TOP item is "1", continue execution with the next instruction.

ULOOP-IF-NOT-CARRY-CLEAR—If the LOOP COUNTER is not "0" and the floating point exponents found in TOP and NEXT are not aligned, continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the exponents are aligned, continue execution with the next instruction. This instruction is specifically designed for combination with special SHIFT instructions to align two floating point numbers.

ULOOP-NEVER—(DECREMENT-LOOP-COUNTER) Decrement the LOOP COUNTER. Continue execution with the next instruction.

ULOOP-IF-NOT-ZERO—If the LOOP COUNTER is not "0" and the TOP item of the Parameter Stack is "0", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the TOP item is "1", continue execution with the next instruction.

ULOOP-IF-NEGATIVE—If the LOOP COUNTER is not "0" and the most significant bit (sign bit) of the TOP item of the Parameter Stack is "1", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the most significant bit of the Parameter Stack is "0", continue execution with the next instruction.

ULOOP-IF-CARRY-SET—If the LOOP COUNTER is not "0" and the exponents of the floating point numbers found in TOP and NEXT are not aligned, continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the exponents are aligned, continue execution with the next instruction.

RETURN FROM SUBROUTINE OR INTERRUPT

Subroutine calls and interrupt acknowledgements cause a redirection of normal program execution. In both cases, the current Program Counter is pushed onto the Return Stack, so the microprocessor can return to its place in the program after executing the subroutine or interrupt service routine.

NOTE: When a CALL to subroutine or interrupt is acknowledged the Program Counter has already been incremented and is pointing to the 4-byte instruction group following the 4-byte group currently being executed. The instruction decoding logic allows the microprocessor to perform a test and execute a return conditional on the outcome of the test in a single cycle. A RETURN pops an address from the Return Stack and stores it to the Program Counter.

RETURN INSTRUCTIONS

RETURN-ALWAYS—Pop the top item from the Return Stack and transfer it to the Program Counter.

RETURN-IF-ZERO—If the TOP item on the Parameter Stack is "0", pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

RETURN-IF-POSITIVE—If the most significant bit (sign bit) of the TOP item on the Parameter Stack is a "0", pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

RETURN-IF-CARRY-CLEAR—If the exponents of the floating point numbers found in TOP and NEXT are not aligned, pop the top item from the Return Stack and

transfer it to the Program Counter Otherwise execute the next instruction.

RETURN-NEVER (NOP)—Execute the next instruction.

RETURN-IF-NOT-ZERO—If the TOP item on the Parameter Stack is not "0", pop the top item from the Return Stack and transfer it to the Program Counter Otherwise execute the next instruction.

RETURN-IF-NEGATIVE—If the most significant bit (sign bit) of the TOP item on the Parameter Stack is a "1", pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

RETURN-IF-CARRY-SET—If the exponents of the floating point numbers found in TOP and NEXT are aligned, pop the top item from the Return Stack and transfer it to the Program Counter Otherwise execute the next instruction.

HANDLING MEMORY FROM DYNAMIC RAM

The microprocessor **50**, like any RISC type architecture, is optimized to handle as many operations as possible on-chip for maximum speed External memory operations take from 80 nsec. to 220 nsec compared with on-chip memory speeds of from 4 nsec. to 30 nsec There are times when external memory must be accessed

External memory is accessed using three registers:

X-REGISTER—A 30-bit memory pointer which can be used for memory access and simultaneously incremented or decremented

Y-REGISTER—A 30-bit memory pointer which can be used for memory access and simultaneously incremented or decremented

PROGRAM-COUNTER—A 30-bit memory pointer normally used to point to 4-byte instruction groups External memory may be accessed at addresses relative to the PC The operands are sometimes called "Immediate" or "Literal" in other computers When used as memory pointer, the PC is also incremented after each operation.

MEMORY LOAD & STORE INSTRUCTIONS

FETCH-VIA-X—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack X is unchanged

FETCH-VIA-Y—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack Y is unchanged

FETCH-VIA-X-AUTOINCREMENT—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack After fetching, increment the most significant 30 bits of X to point to the next 32-bit word address

FETCH-VIA-Y-AUTOINCREMENT—Fetch the 32-bit memory content pointed to by Y and push it onto the Parameter Stack After fetching, increment the most significant 30 bits of Y to point to the next 32-bit word address.

FETCH-VIA-X-AUTODECREMENT—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack After fetching, decrement the most significant 30 bits of X to point to the previous 32-bit word address

FETCH-VIA-Y-AUTODECREMENT—Fetch the 32-bit memory content pointed to by Y and push it onto the Parameter Stack After fetching decrement the most significant 30 bits of Y to point to the previous 32-bit word address.

STORE-VIA-X—Pop the top item of the Parameter Stack and store it in the memory location pointed to by X X is unchanged

STORE-VIA-Y—Pop the top item of the Parameter Stack and store it in the memory location pointed to by Y Y is unchanged

STORE-VIA-X-AUTOINCREMENT—Pop the top item of the Parameter Stack and store it in the memory location pointed to by X After storing, increment the most significant 30 bits of X to point to the next 32-bit word address.

STORE-VIA-Y-AUTOINCREMENT—Pop the top item of the Parameter Stack and store it in the memory location pointed to by Y After storing, increment the most significant 30 bits of Y to point to the next 32-bit word address.

STORE-VIA-X-AUTODECREMENT—Pop the top item of the Parameter Stack and store it in the memory location pointed to by X After storing, decrement the most significant 30 bits of X to point to the previous 32-bit word address.

STORE-VIA-Y-AUTODECREMENT—Pop the top item of the Parameter Stack and store it in the memory location pointed to by Y After storing, decrement the most significant 30 bits of Y to point to the previous 32-bit word address.

FETCH-VIA-PC—Fetch the 32-bit memory content pointed to by the Program Counter and push it onto the Parameter Stack After fetching, increment the most significant 30 bits of the Program Counter to point to the next 32-bit word address.

*NOTE When this instruction executes, the PC is pointing to the memory location following the instruction The effect is of loading a 32-bit immediate operand This is an 8-bit instruction and therefore will be combined with other 8-bit instructions in a 4-byte instruction fetch. It is possible to have from one to four FETCH-VIA-PC instructions in a 4-byte instruction fetch The PC increments after each execution of FETCH-VIA-PC so it is possible to push four immediate operands on the stack The four operands would be the found in the four memory locations following the instruction.

BYTE-FETCH-VIA-X—Fetch the 32-bit memory content pointed to by the most significant 30 bits of X Using the two least significant bits of X, select one of four bytes from the 32-bit memory fetch right justify the byte in a 32-bit field and push the selected byte preceded by leading zeros onto the Parameter Stack

BYTE-STORE-VIA-X—Fetch the 32-bit memory content pointed to by the most significant 30 bits of X Pop the TOP item from the Parameter Stack Using the two least significant bits of X place the least significant byte into the 32-bit memory data and write the 32-bit entity back to the location pointed to by the most significant 30 bits of X

OTHER EFFECTS OF MEMORY ACCESS INSTRUCTIONS:

Any FETCH instruction will push a value on the Parameter Stack **74** If the on-chip stack is full, the stack will overflow into off-chip memory stack resulting in an additional memory cycle Any STORE instruction will pop a value from the Parameter Stack **74** If the on-chip stack is empty, a memory cycle will be generated to fetch a value from off-chip memory stack.

HANDLING ON-CHIP VARIABLES

High-level languages often allow the creation of LOCAL VARIABLES These variables are used by a particular procedure and discarded In cases of nested procedures, layers of these variables must be maintained On-chip storage is up to five times faster than off-chip RAM, so a means of keeping local variables on-chip can make operations run faster The microprocessor **50** provides the capability for both on-chip storage of local variables and nesting of multiple levels of variables through the Return Stack

The Return Stack 134 is implemented as 16 on-chip RAM locations. The most common use for the Return Stack 134 is storage of return addresses from subroutines and interrupt calls. The microprocessor allows these 16 locations to also be used as addressable registers. The 16 locations may be read and written by two instructions which indicate a Return Stack relative address from 0–15. When high-level procedures are nested, the current procedure variables push the previous procedure variables further down the Return Stack 134. Eventually, the Return Stack will automatically overflow into off-chip RAM.

ON-CHIP VARIABLE INSTRUCTIONS

READ-LOCAL-VARIABLE XXXX—Read the XXXXth location relative to the top of the Return Stack. (XXXX is a binary number from 0000–1111). Push the item read onto the Parameter Stack.
OTHER EFFECTS: If the Parameter Stack is full, the push operation will cause a memory cycle to be generated as one item of the stack is automatically stored to external RAM. The logic which selects the location performs a modulo 16 subtraction. If four local variables have been pushed onto the Return Stack, and an instruction attempts to READ the fifth item, unknown data will be returned.

WRITE-LOCAL-VARIABLE XXXX—Pop the TOP item of the Parameter Stack and write it into the XXXXth location relative to the top of the Return Stack. (XXXX is a binary number from 0000–1111.)
OTHER EFFECTS: If the Parameter Stack is empty, the pop operation will cause a memory cycle to be generated to fetch the Parameter Stack item from external RAM. The logic which selects the location performs a modulo 16 subtraction. If four local variables have been pushed onto the Return Stack, and an instruction attempts to WRITE to the fifth item, it is possible to clobber return addresses or wreak other havoc.

REGISTER AND FLIP-FLOP TRANSFER AND PUSH INSTRUCTIONS

DROP—Pop the TOP item from the Parameter Stack and discard it.

SWAP—Exchange the data in the TOP Parameter Stack location with the data in the NEXT Parameter Stack location.

DUP—Duplicate the TOP item on the Parameter Stack and push it onto the Parameter Stack.

PUSH-LOOP-COUNTER—Push the value in LOOP COUNTER onto the Parameter Stack.

POP-RSTACK-PUSH-TO-STACK—Pop the top item from the Return Stack and push it onto the Parameter Stack.

PUSH-X-REG—Push the value in the X Register onto the Parameter Stack.

PUSH-STACK-POINTER—Push the value of the Parameter Stack pointer onto the Parameter Stack.

PUSH-RSTACK-POINTER—Push the value of the Return Stack pointer onto the Return Stack.

PUSH-MODE-BITS—Push the value of the MODE REGISTER onto the Parameter Stack.

PUSH-INPUT—Read the 10 dedicated input bits and push the value (right justified and padded with leading zeros) onto the Parameter Stack.

SET-LOOP-COUNTER—Pop the TOP value from the Parameter Stack and store it into LOOP COUNTER.

POP-STACK-PUSH-TO-RSTACK—Pop the TOP item from the Parameter Stack and push it onto the Return Stack.

SET-X-REG—Pop the TOP item from the Parameter Stack and store it into the X Register.

SET-STACK-POINTER—Pop the TOP item from the Parameter Stack and store it into the Stack Pointer.

SET-RSTACK-POINTER—Pop the TOP item from the Parameter Stack and store it into the Return Stack Pointer.

SET-MODE-BITS—Pop the TOP value from the Parameter Stack and store it into the MODE BITS.

SET-OUTPUT—Pop the TOP item from the Parameter Stack and output it to the 10 dedicated output bits.
OTHER EFFECTS: Instructions which push or pop the Parameter Stack or Return Stack may cause a memory cycle as the stacks overflow back and forth between on-chip and off-chip memory.

LOADING A SHORT LITERAL

A special case of register transfer instruction is used to push an 8-bit literal onto the Parameter Stack. This instruction requires that the 8-bits to be pushed reside in the last byte of a 4-byte instruction group. The instruction op-code loading the literal may reside in ANY of the other three bytes in the instruction group.

EXAMPLE

BYTE 1	BYTE 2	BYTE 3
LOAD-SHORT-LITERAL	QQQQQQQQ	QQQQQQQQ
BYTE 4		
00001111		

In this example, QQQQQQQQ indicates any other 8-bit instruction. When Byte 1 is executed, binary 00001111(HEX 0f) from Byte 4 will be pushed (right justified and padded by leading zeros) onto the Parameter Stack. Then the instructions in Byte 2 and Byte 3 will execute. The microprocessor instruction decoder knows not to execute Byte 4. It is possible to push three identical 8-bit values as follows:

BYTE 1	BYTE 2
LOAD-SHORT-LITERAL	LOAD-SHORT-LITERAL
BYTE 3	BYTE 4
LOAD-SHORT-LITERAL	00001111
SHORT-LITERAL-INSTRUCTION	

LOAD-SHORT-LITERAL—Push the 8-bit value found in Byte 4 of the current 4-byte instruction group onto the Parameter Stack.

LOGIC INSTRUCTIONS

Logical and math operations used the stack for the source of one or two operands and as the destination for results. The stack organization is a particularly convenient arrangement for evaluating expressions. TOP indicates the top value on the Parameter Stack 74. NEXT indicates the next to top value on the Parameter Stack 74.

AND—Pop TOP and NEXT from the Parameter Stack, perform the logical AND operation on these two operands, and push the result onto the Parameter Stack.

OR—Pop TOP and NEXT from the Parameter Stack, perform the logical OR operation on these two operands, and push the result onto the Parameter Stack.

XOR—Pop TOP and NEXT from the Parameter Stack, perform the logical exclusive OR on these two operands, and push the result onto the Parameter Stack.

BIT-CLEAR—Pop TOP and NEXT from the Parameter Stack, toggle all bits in NEXT, perform the logical AND operation on TOP, and push the result onto the Parameter Stack. (Another way of understanding this instruction is thinking of it as clearing all bits in TOP that are set in NEXT.)

MATH INSTRUCTIONS

Math instruction pop the TOP item and NEXT to top item of the Parameter Stack 74 to use as the operands. The results are pushed back on the Parameter Stack. The CARRY flag is used to latch the "33rd bit" of the ALU result.

ADD—Pop the TOP item and NEXT to top item from the Parameter Stack, add the values together and push the result back on the Parameter Stack. The CARRY flag may be changed.

ADD-WITH-CARRY—Pop the TOP item and the NEXT to top item from the Parameter Stack, add the values together. If the CARRY flag is '1' increment the result. Push the ultimate result back on the Parameter Stack. The CARRY flag may be changed.

ADD-X—Pop the TOP item from the Parameter Stack and read the third item from the top of the Parameter Stack. Add the values together and push the result back on the Parameter Stack. The CARRY flag may be changed.

SUB—Pop the TOP item and NEXT to top item from the Parameter Stack, Subtract NEXT from TOP and push the result back on the Parameter Stack. The CARRY flag may be changed.

SUB-WITH-CARRY—Pop the TOP item and NEXT to top item from the Parameter Stack. Subtract NEXT from TOP. If the CARRY flag is '1' increment the result. Push the ultimate result back on the Parameter Stack. The CARRY flag may be changed.

SUB-X—

SIGNED-MULT-STEP—

UNSIGNED-MULT-STEP—

SIGNED-FAST-MULT—

FAST-MULT-STEP—

UNSIGNED-DIV-STEP—

GENERATE-POLYNOMIAL—

ROUND—

COMPARE—Pop the TOP item and NEXT to top item from the Parameter Stack. Subtract NEXT from TOP. If the result has the most significant bit equal to "0" (the result is positive), push the result onto the Parameter Stack. If the result has the most significant bit equal to '1' (the result is negative), push the old value of TOP onto the Parameter Stack. The CARRY flag may be affected.

SHIFT/ROTATE

SHIFT-LEFT—Shift the TOP Parameter Stack item left one bit. The CARRY flag is shifted into the least significant bit of TOP.

SHIFT-RIGHT—Shift the TOP Parameter Stack item right one bit. The least significant bit of TOP is shifted into the CARRY flag. Zero is shifted into the most significant bit of TOP.

DOUBLE-SHIFT-LEFT—Treating the TOP item of the Parameter Stack as the most significant word of a 64-bit number and the NEXT stack item as the least significant word, shift the combined 64-bit entity left one bit. The CARRY flag is shifted into the least significant bit of NEXT.

DOUBLE-SHIFT-RIGHT—Treating the TOP item of the Parameter Stack as the most significant word of a 64-bit number and the NEXT stack item as the least significant word, shift the combined 64-bit entity right one bit. The least significant bit of NEXT is shifted into the CARRY flag. Zero is shifted into the most significant bit of TOP.

OTHER INSTRUCTIONS

FLUSH-STACK—Empty all on-chip Parameter Stack locations into off-chip RAM. (This instruction is useful for multitasking applications). This instruction accesses a counter which holds the depth of the on-chip stack and can require from none to 16 external memory cycles.

FLUSH-RSTACK—Empty all on-chip Return Stack locations into off-chip RAM. (This instruction is useful for multitasking applications). This instruction accesses a counter which holds the depth of the on-chip Return Stack and can require from none to 16 external memory cycles.

It should further be apparent to those skilled in the art that various changes in form and details of the invention as shown and described may be made. It is intended that such changes be included within the spirit and scope of the claims appended hereto.

What is claimed is:

1. A microprocessor system, comprising a single integrated circuit including a central processing unit and an entire ring oscillator variable speed system clock in said single integrated circuit and connected to said central processing unit for clocking said central processing unit, said central processing unit and said ring oscillator variable speed system clock each including a plurality of electronic devices correspondingly constructed of the same process technology with corresponding manufacturing variations, a processing frequency capability of said central processing unit and a speed of said ring oscillator variable speed system clock varying together due to said manufacturing variations and due to at least operating voltage and temperature of said single integrated circuit; an on-chip input/output interface connected to exchange coupling control signals, addresses and data with said central processing unit; and a second clock independent of said ring oscillator variable speed system clock connected to said input/output interface.

2. The microprocessor system of claim 1 in which said second clock is a fixed frequency clock.

3. In a microprocessor integrated circuit, a method for clocking the microprocessor within the integrated circuit comprising the steps of:

providing an entire ring oscillator system clock constructed of electronic devices within the integrated circuit, said electronic devices having operating characteristics which will because said entire ring oscillator system clock and said microprocessor are located within the same integrated circuit vary together with operating characteristics of electronic devices included within the microprocessor;

using the ring oscillator system clock for clocking the microprocessor, said microprocessor operating at a variable processing frequency dependent upon a variable speed of said ring oscillator system clock;

providing an on chip input/output interface for the microprocessor integrated circuit; and

clocking the input/output interface with a second clock independent of the ring oscillator system clock.

4. The method of claim 3 in which the second clock is a fixed frequency clock.

5. The method of claim 3 further including the step of:

transferring information to and from said microprocessor in synchrony with said ring oscillator system clock.

6. A microprocessor system comprising:

a central processing unit disposed upon an integrated circuit substrate, said central processing unit operating at a processing frequency and being constructed of a first plurality of electronic devices;

an entire oscillator disposed upon said integrated circuit substrate and connected to said central processing unit, said oscillator clocking said central processing unit at a clock rate and being constructed of a second plurality of electronic devices, thus varying the processing frequency of said first plurality of electronic devices and

the clock rate of said second plurality of electronic devices in the same way as a function of parameter variation in one or more fabrication or operational parameters associated with said integrated circuit substrate, thereby enabling said processing frequency to track said clock rate in response to said parameter variation;

an on-chip input output interface, connected between said said central processing unit and an external memory bus, for facilitating exchanging coupling control signals, addresses and data with said central processing unit; and

an external clock, independent of said oscillator connected to said input/output interface wherein said external clock is operative at a frequency independent of a clock frequency of said oscillator

7 The microprocessor system of claim **6** wherein said one or more operational parameters include operating temperature of said substrate or operating voltage of said substrate.

8 The microprocessor system of claim **6** wherein said external clock comprises a fixed-frequency clock which operates synchronously relative to said oscillator

9 The microprocessor system of claim **6** wherein said oscillator comprises a ring oscillator

10 In a microprocessor system including a central processing unit, a method for clocking said central processing unit comprising the steps of:

providing said central processing unit upon an integrated circuit substrate, said central processing unit being constructed of a first plurality of transistors and being operative at a processing frequency;

providing an entire variable speed clock disposed upon said integrated circuit substrate, said variable speed clock being constructed of a second plurality of transistors;

clocking said central processing unit at a clock rate using said variable speed clock with said central processing unit being clocked by said variable speed clock at a variable frequency dependent upon variation in one or more fabrication or operational parameters associated with said integrated circuit substrate, said processing frequency and said clock rate varying in the same way relative to said variation in said one or more fabrication or operational parameters associated with said integrated circuit substrate;

connecting an on chip input/output interface between said central processing unit and an external memory bus, and exchanging coupling control signals, addresses and data between said input/output interface and said central processing unit; and

clocking said input/output interface using an external clock wherein said external clock is operative at a frequency independent of a clock frequency of said oscillator

* * * * *

US005440749A

United States Patent [19]

Moore et al.

[11] Patent Number: 5,440,749

[45] Date of Patent: Aug. 8, 1995

[54] HIGH PERFORMANCE, LOW COST MICROPROCESSOR ARCHITECTURE

[75] Inventors: Charles H. Moore, Woodside; Russell H. Fish, III, Mt. View, both of Calif.

[73] Assignee: Nanotronics Corporation, Eagle Point, Oreg.

[21] Appl. No.: 389,334

[22] Filed: Aug. 3, 1989

[51] Int. Cl.[6] G06F 9/22

[52] U.S. Cl. 395/800; 364/931; 364/925.6; 364/937.1; 364/965.4; 364/232.8; 364/244.3

[58] Field of Search 395/425, 725, 775, 800

[56] References Cited

U.S. PATENT DOCUMENTS

3,603,934	9/1971	Heath	364/DIG. 1
4,003,033	1/1977	O'Keefe et al.	364/200
4,037,090	7/1977	Raymond	364/200
4,042,972	8/1977	Grunes et al.	364/200
4,050,058	9/1977	Garlic	395/800
4,067,059	1/1978	Derchak	364/DIG. 1
4,079,455	3/1978	Ozga	395/800
4,110,822	8/1978	Porter	364/200
4,125,871	11/1978	Martin	364/DIG. 2
4,128,873	12/1978	Lamiaux	364/200
4,255,785	3/1981	Chamberlin	395/375
4,354,228	10/1982	Moore et al.	364/200
4,376,977	3/1983	Brunshorst	364/DIG. 1
4,382,279	5/1983	Mgon	364/200
4,403,303	9/1983	Howes et al.	364/900
4,450,519	5/1984	Guttag et al.	364/200
4,463,421	7/1984	Laws	395/325
4,538,239	8/1985	Magar	364/759
4,541,045	9/1985	Kromer	395/375
4,562,537	12/1985	Barnett et al.	395/375
4,577,282	3/1986	Caudel et al.	395/800
4,607,332	8/1986	Goldberg	364/900
4,626,988	12/1986	George et al.	364/200
4,649,471	3/1987	Briggs	395/325
4,665,495	5/1987	Thaden	345/185
4,709,329	11/1987	Hecker	395/275
4,713,749	12/1987	Magar et al.	395/375
4,714,994	12/1987	Oklobdzija et al.	395/375
4,720,812	1/1988	Kao et al.	395/700
4,772,888	9/1988	Kimura	340/825.5
4,777,591	10/1988	Chang et al.	395/800
4,787,032	11/1988	Culley et al.	364/200
4,803,621	2/1989	Kelly	395/400
4,860,198	8/1989	Takenaka	364/DIG. 1
4,870,562	9/1989	Kimoto	364/DIG. 1
4,931,986	6/1990	Daniel et al.	395/550
5,036,460	7/1991	Takahira	395/425
5,070,451	12/1991	Moore et al.	395/375
5,127,091	6/1992	Bonfarah	395/375

OTHER PUBLICATIONS

Intel 80386 Programmer's Reference Manual, 1986.

Primary Examiner—David Y. Eng
Attorney, Agent, or Firm—Cooley Godward Castro Huddleson & Tatum

[57] ABSTRACT

A microprocessor (50) includes a main central processing unit (CPU) (70) and a separate direct memory access (DMA) CPU (72) in a single integrated circuit making up the microprocessor (50). The main CPU (70) has a first 16 deep push down tack (74), which has a to item register (76) and a next item register (78), respectively connected to provide inputs to an arithmetic logic unit (ALU) (80) by lines (82) and (84). An output of the ALU (80) is connected to the top item register at (82) is also connected by line (88) to an internal data bus (90). CPU (70) is pipeline free. The simplified CPU (70) requires fewer transistors to implement than pipelined architectures, yet produces performance which matches or exceeds existing techniques. The DMA CPU (72) provides inputs to the memory controller (118) on line (148). The memory controller (118) is connected to a RAM by address/data bus (150) and control lines (152). The DMA CPU (72) enables the CPU (70) to execute instructions four times faster than the RAM speed by fetching four instructions in a single memory cycle.

29 Claims, 19 Drawing Sheets

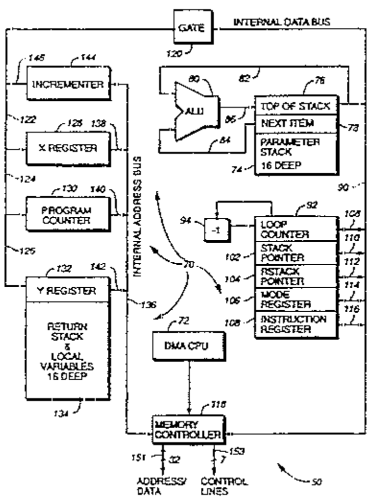


EXHIBIT B

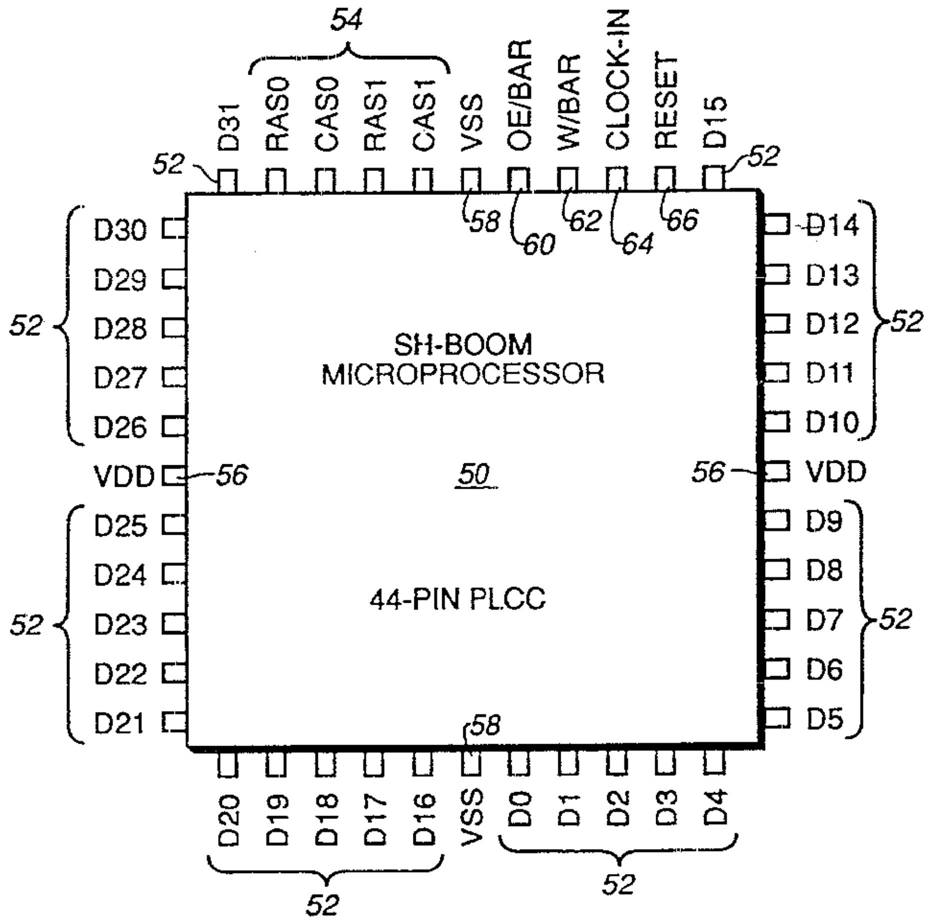
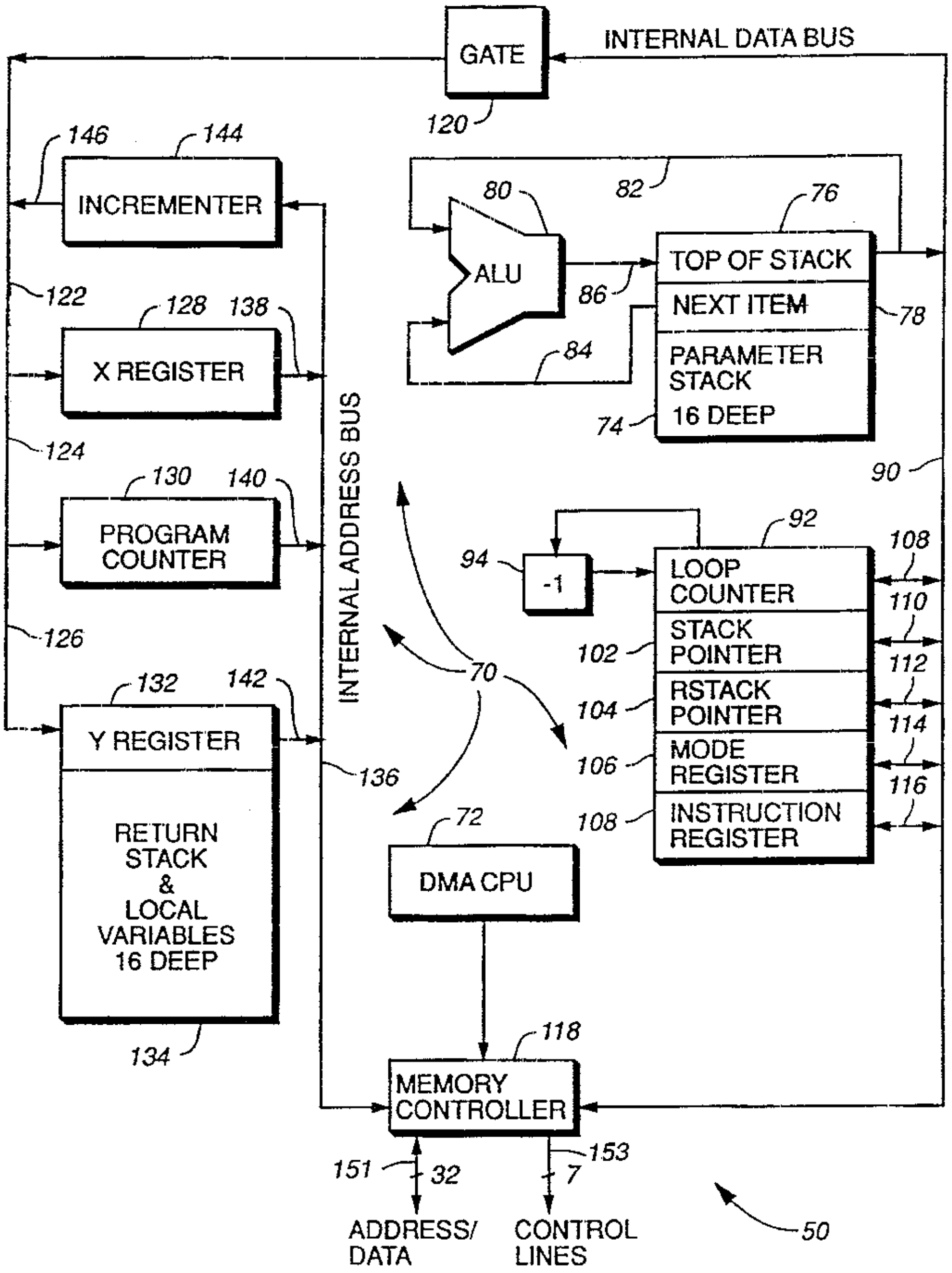

54
D31
RAS0
CAS0
RAS1
CAS1
VSS
OE/BAR
W/BAR
CLOCK-IN
RESET
D15
52
58
60
62
64
66
D30
D29
D28
D27
D26
VDD
56
D25
D24
D23
D22
D21
D14
D13
D12
D11
D10
D9
D8
D7
D6
D5
SH-BOOM
MICROPROCESSOR
50
44-PIN PLCC
D20
D19
D18
D17
D16
D0
D1
D2
D3
D4

FIG._1



GATE
INTERNAL DATA BUS
120
146
144
INCREMENTER
80
82
76
ALU
TOP OF STACK
86
NEXT ITEM
78
122
128
138
X REGISTER
84
PARAMETER STACK
16 DEEP
74
INTERNAL ADDRESS BUS
124
130
140
PROGRAM COUNTER
90
92
108
94
-1
LOOP COUNTER
110
126
STACK POINTER
102
112
70
RSTACK POINTER
104
114
132
142
Y REGISTER
MODE REGISTER
106
116
136
72
108
INSTRUCTION REGISTER
RETURN STACK & LOCAL VARIABLES 16 DEEP
DMA CPU
118
134
MEMORY CONTROLLER
151
32
153
7
ADDRESS/ DATA
CONTROL LINES
50

FIG._2

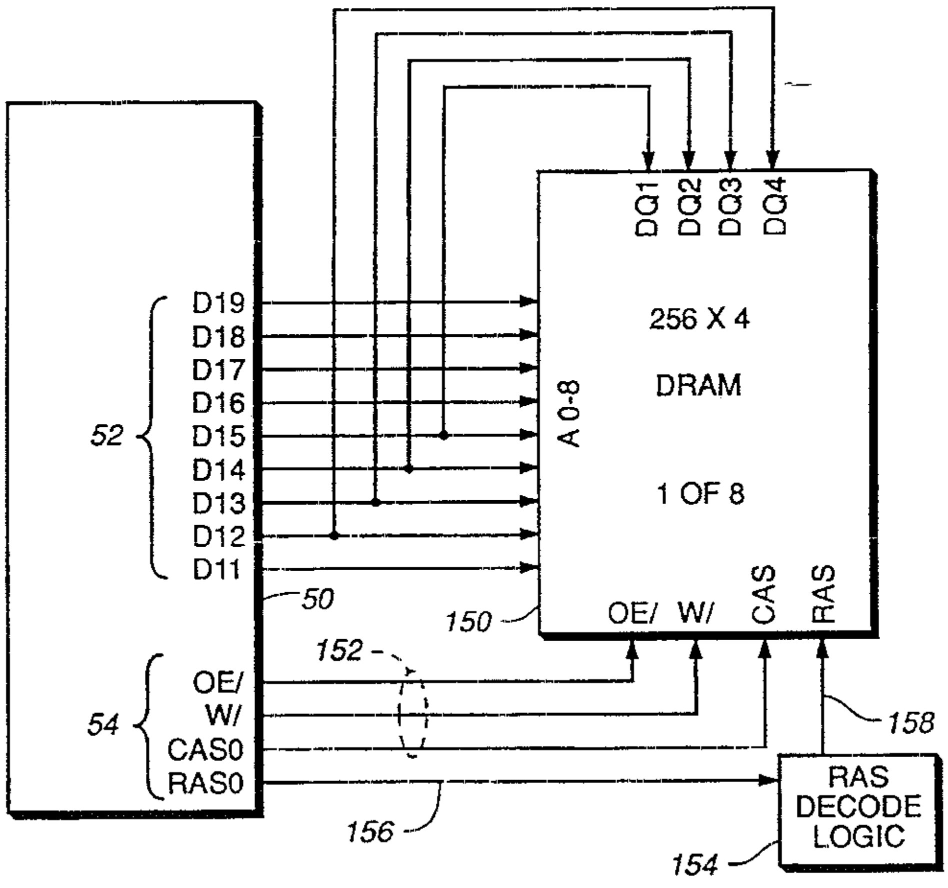

DQ1
DQ2
DQ3
DQ4
256 X 4
DRAM
1 OF 8
A 0-8
D19
D18
D17
D16
D15
D14
D13
D12
D11
52
50
150
OE/
W/
CAS
RAS
152
54
OE/
W/
CAS0
RAS0
158
RAS
DECODE
LOGIC
156
154

FIG._3

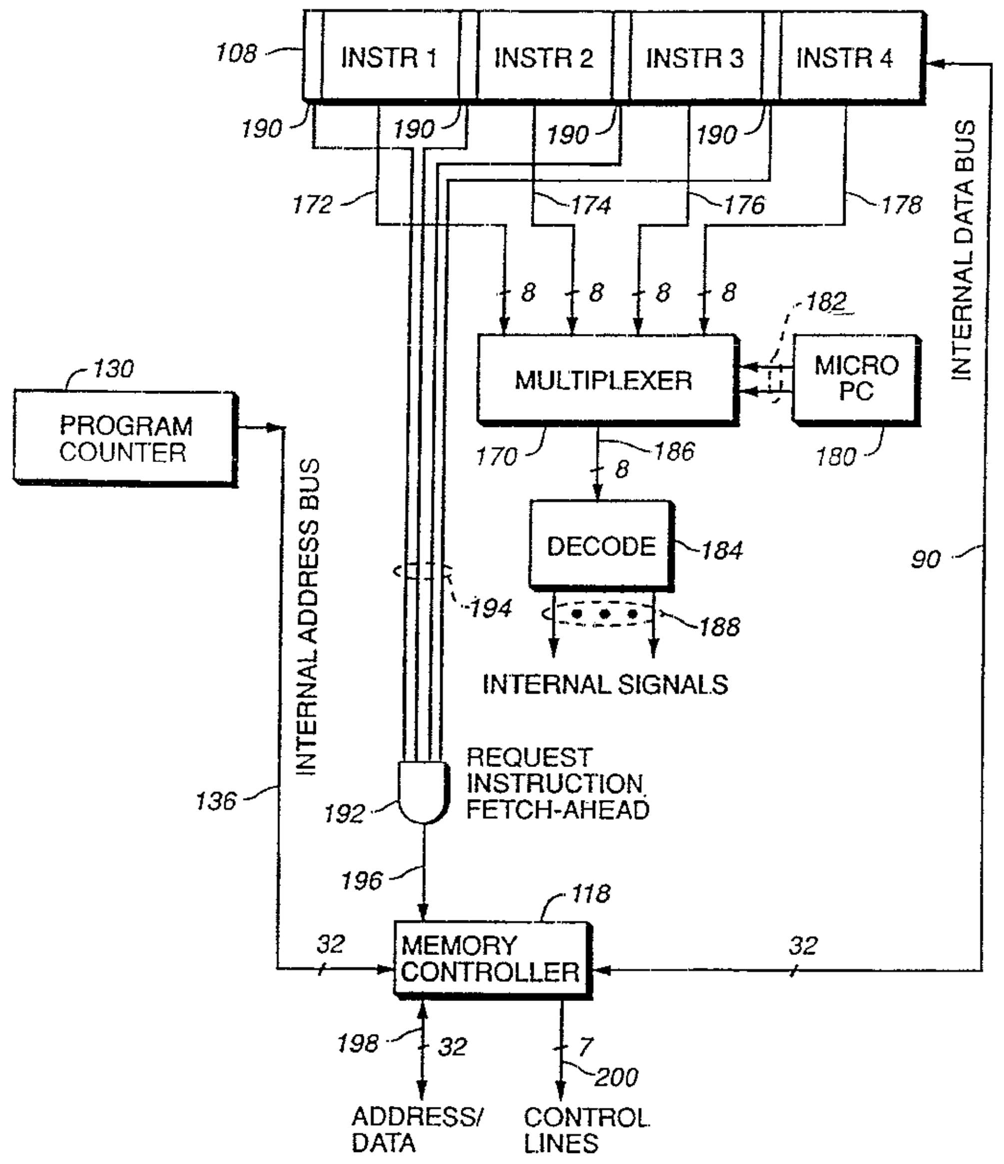

108
INSTR 1
INSTR 2
INSTR 3
INSTR 4
190
190
190
190
172
174
176
178
8
8
8
8
182
MULTIPLEXER
MICRO PC
130
PROGRAM COUNTER
INTERNAL DATA BUS
INTERNAL ADDRESS BUS
170
186
180
8
DECODE
184
90
194
188
INTERNAL SIGNALS
REQUEST INSTRUCTION FETCH-AHEAD
136
192
196
118
32
MEMORY CONTROLLER
32
198
32
7
200
ADDRESS/ DATA
CONTROL LINES

FIG._4

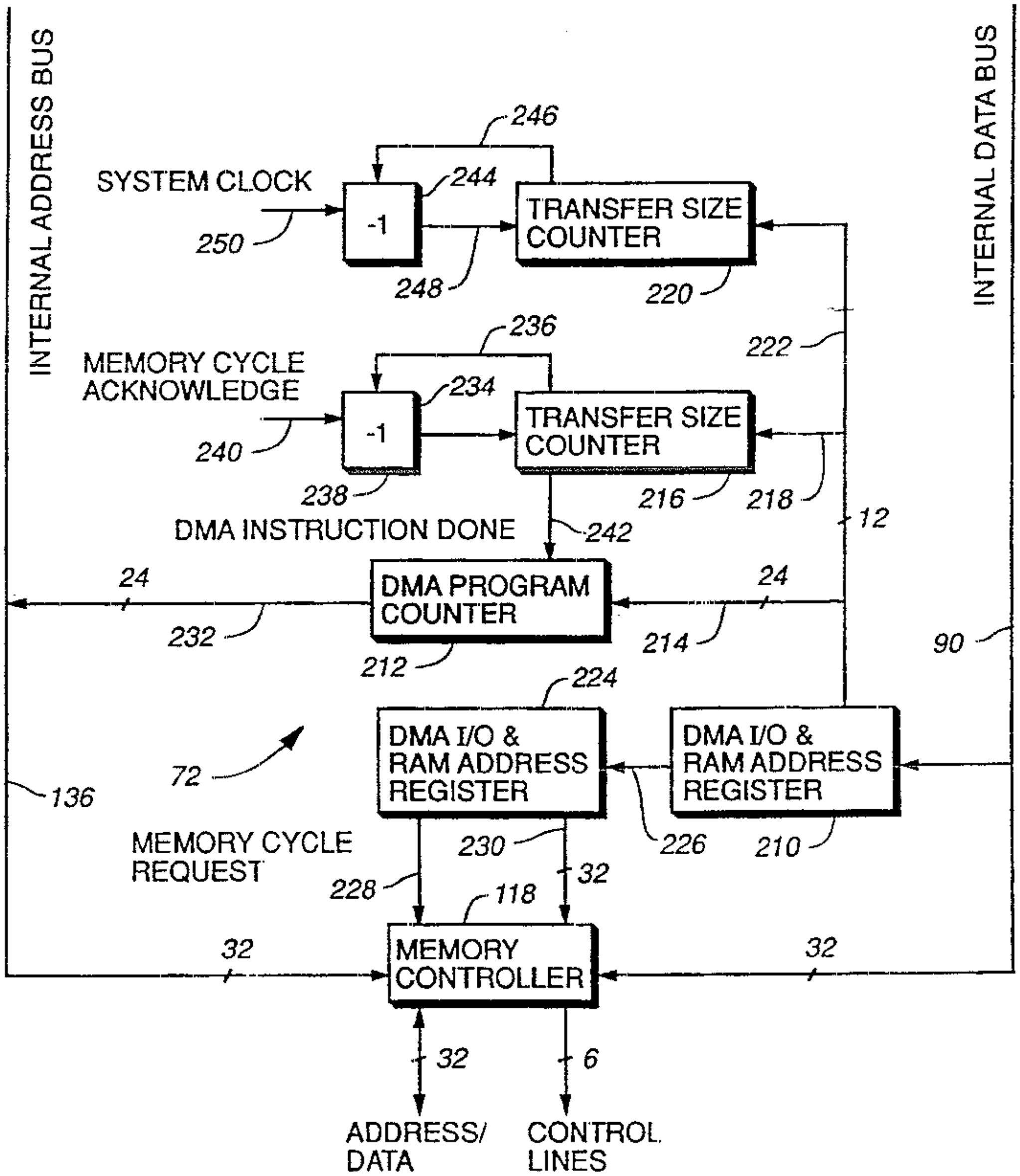
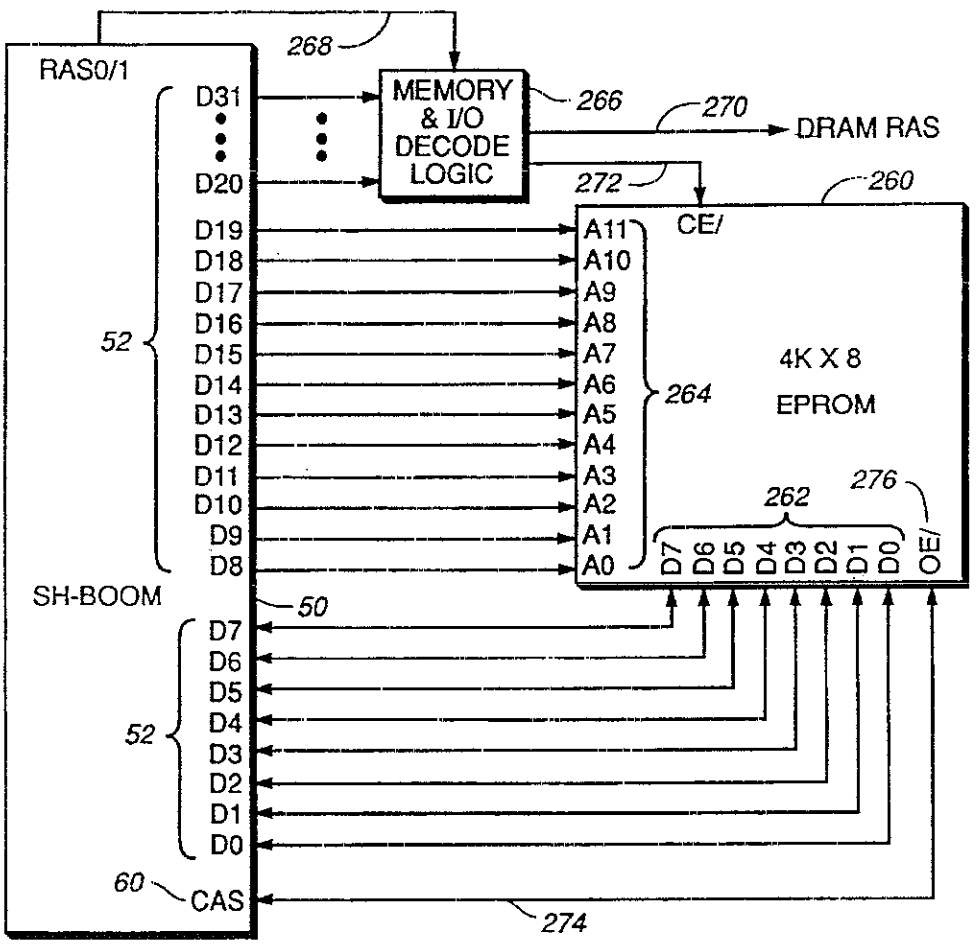
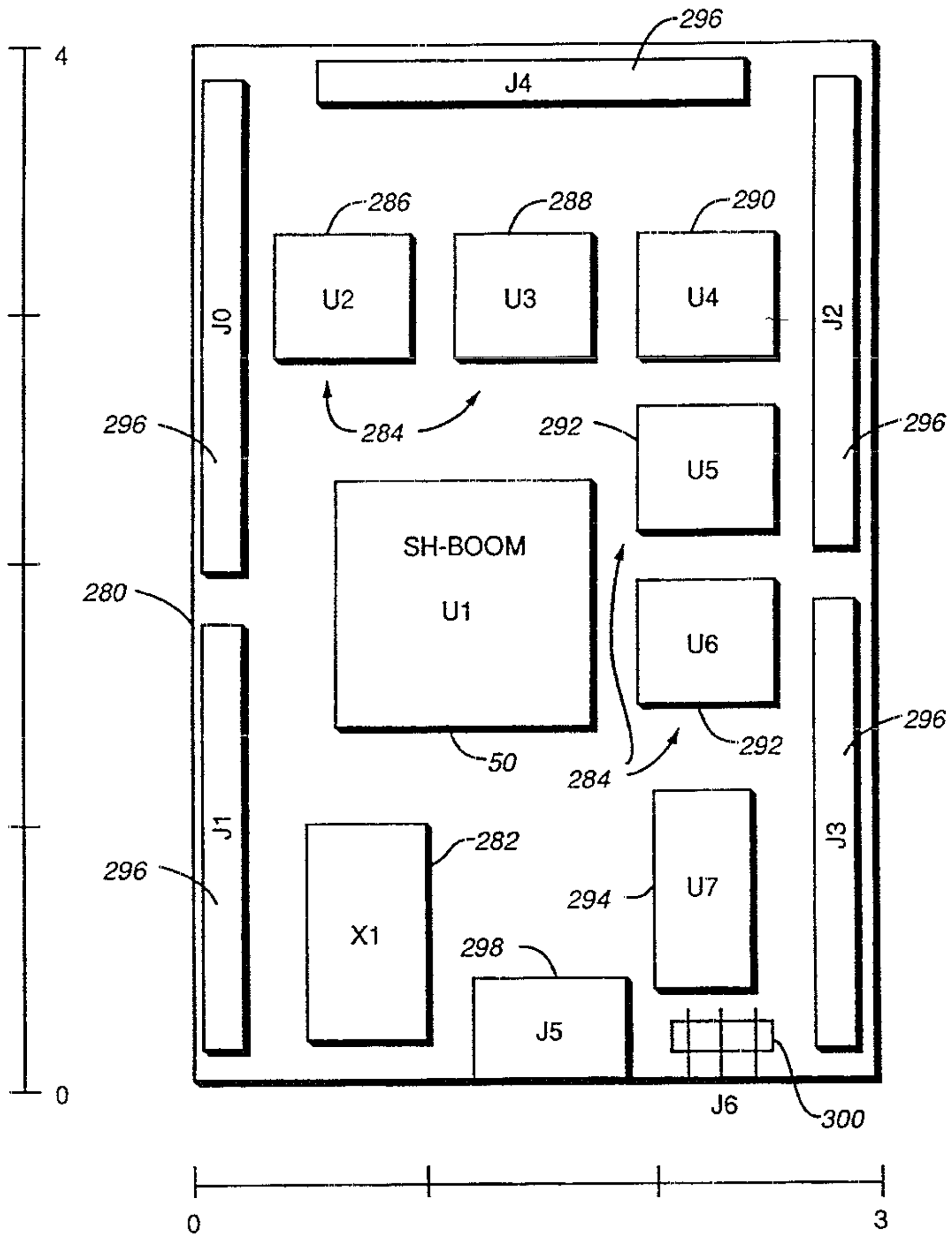

INTERNAL ADDRESS BUS
INTERNAL DATA BUS
SYSTEM CLOCK
250
-1
244
246
248
TRANSFER SIZE COUNTER
220
222
MEMORY CYCLE ACKNOWLEDGE
240
-1
234
236
238
TRANSFER SIZE COUNTER
216
218
12
DMA INSTRUCTION DONE
242
24
DMA PROGRAM COUNTER
232
212
24
214
90
224
DMA I/O & RAM ADDRESS REGISTER
DMA I/O & RAM ADDRESS REGISTER
72
136
226
210
MEMORY CYCLE REQUEST
228
230
32
118
32
MEMORY CONTROLLER
32
32
6
ADDRESS/ DATA
CONTROL LINES

FIG._5



RAS0/1
268
MEMORY & I/O DECODE LOGIC
266
270
DRAM RAS
272
260
D31
D20
D19
D18
D17
D16
D15
D14
D13
D12
D11
D10
D9
D8
52
CE/
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
264
4K X 8
EPROM
262
276
D7 D6 D5 D4 D3 D2 D1 D0 OE/
SH-BOOM
50
D7
D6
D5
D4
D3
D2
D1
D0
52
60
CAS
274

FIG._6



296
4
J4
J0
J2
286
288
290
U2
U3
U4
296
284
292
296
U5
SH-BOOM
U1
280
U6
50
284
292
296
J1
J3
X1
282
296
294
U7
298
J5
J6
300
0
0
3

FIG._7

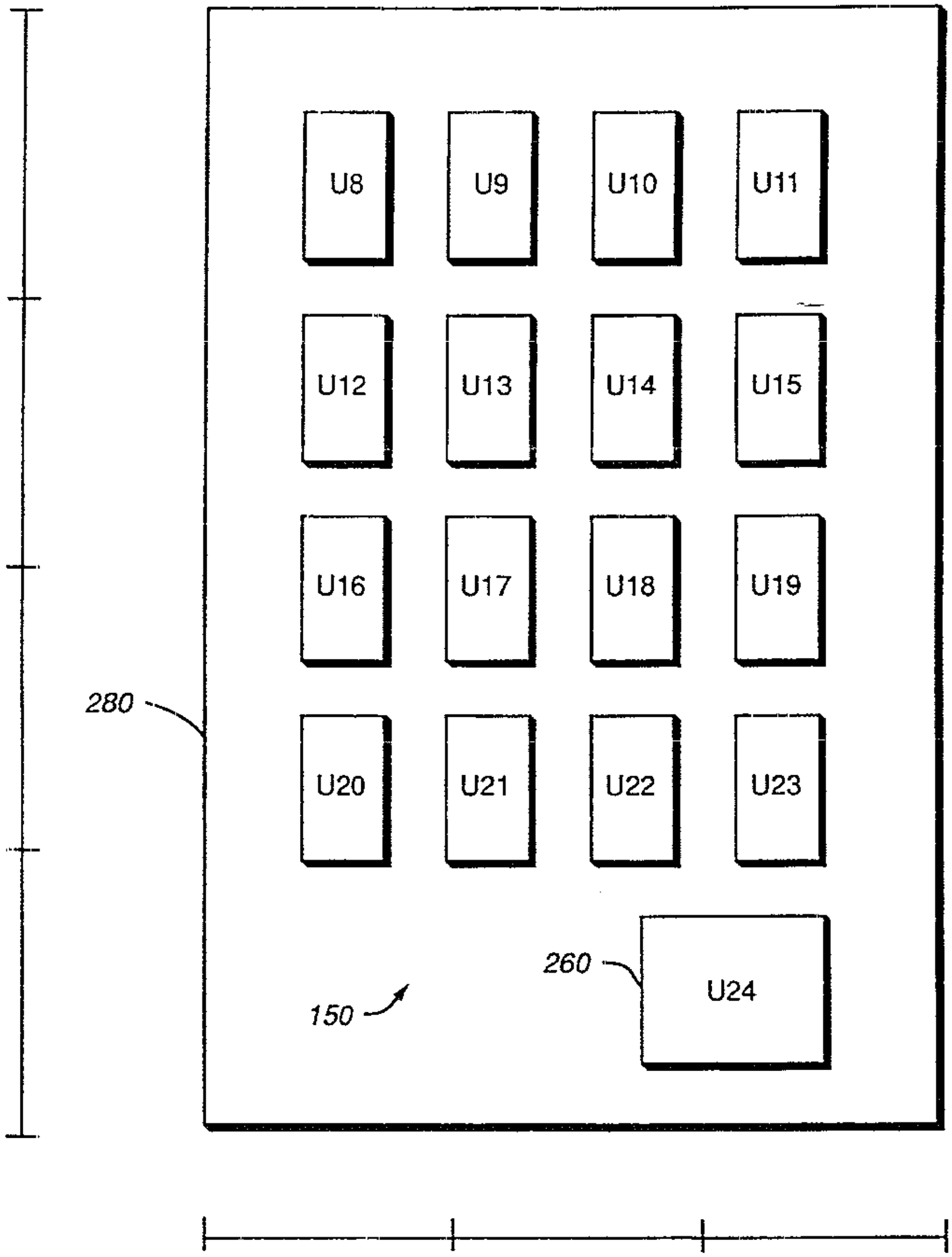
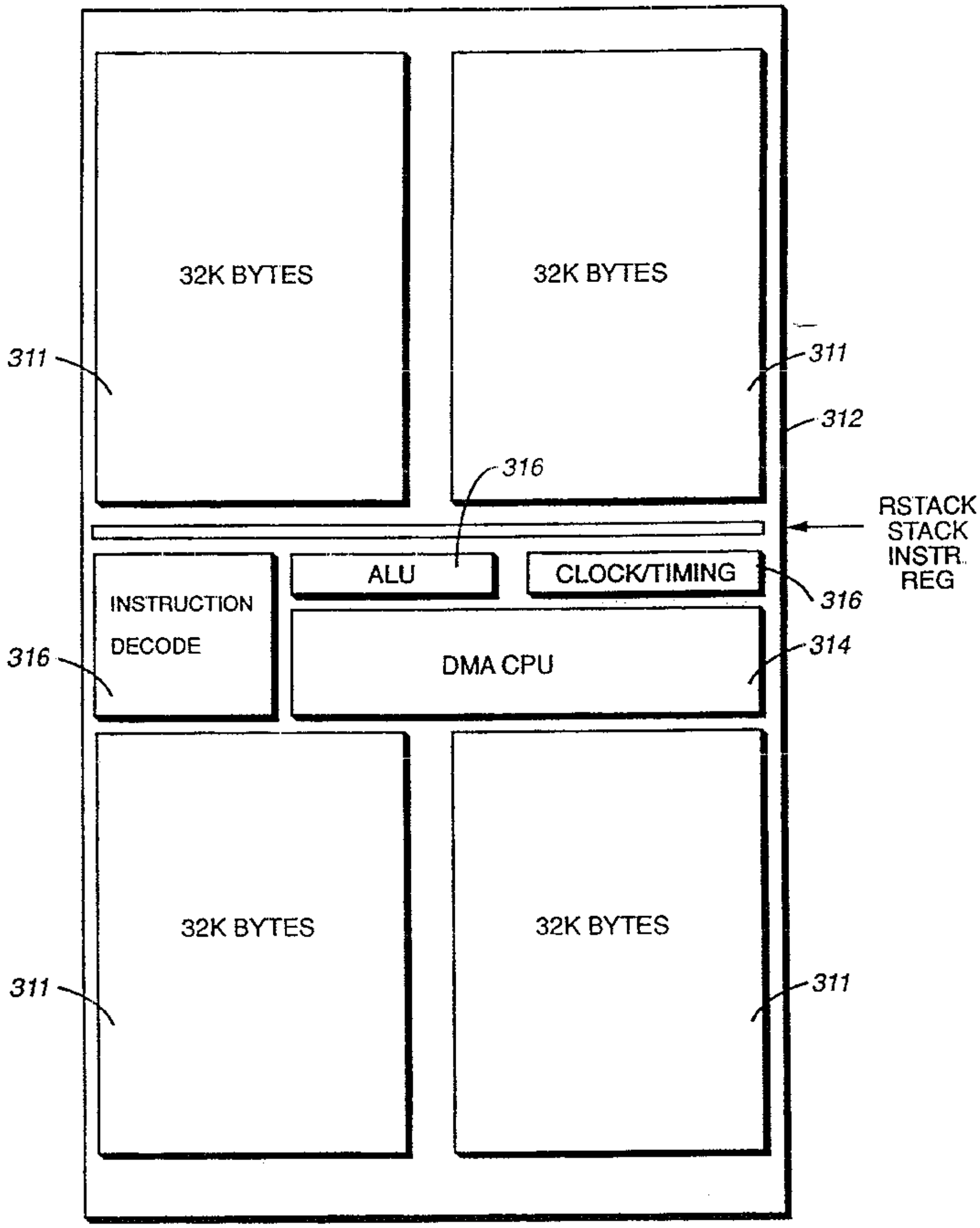

U8
U9
U10
U11
U12
U13
U14
U15
U16
U17
U18
U19
U20
U21
U22
U23
U24
280
260
150

FIG._8



32K BYTES
32K BYTES
311
311
312
316
RSTACK
STACK
INSTR.
REG
ALU
CLOCK/TIMING
316
INSTRUCTION
DECODE
316
DMA CPU
314
32K BYTES
32K BYTES
311
311

FIG._9

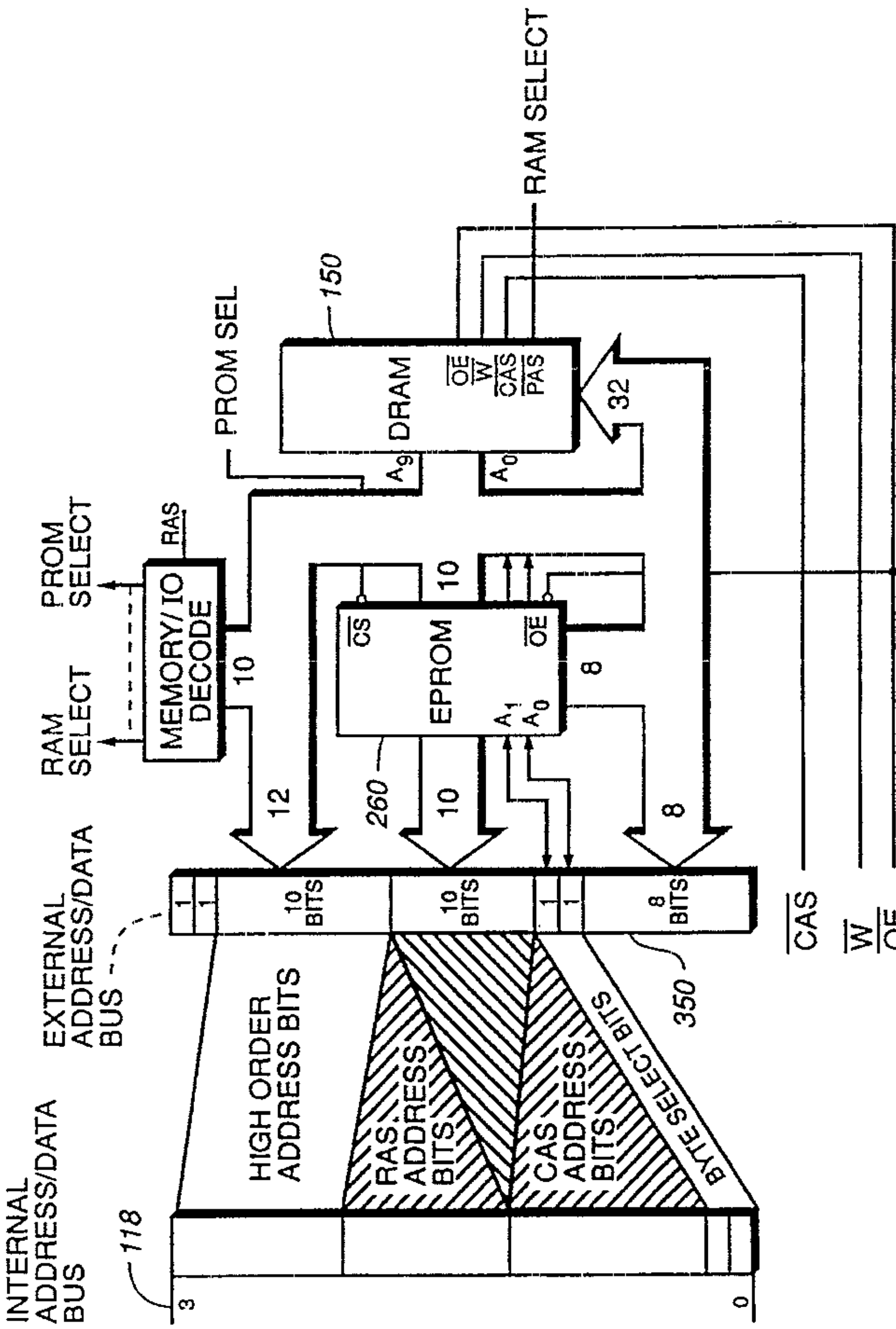
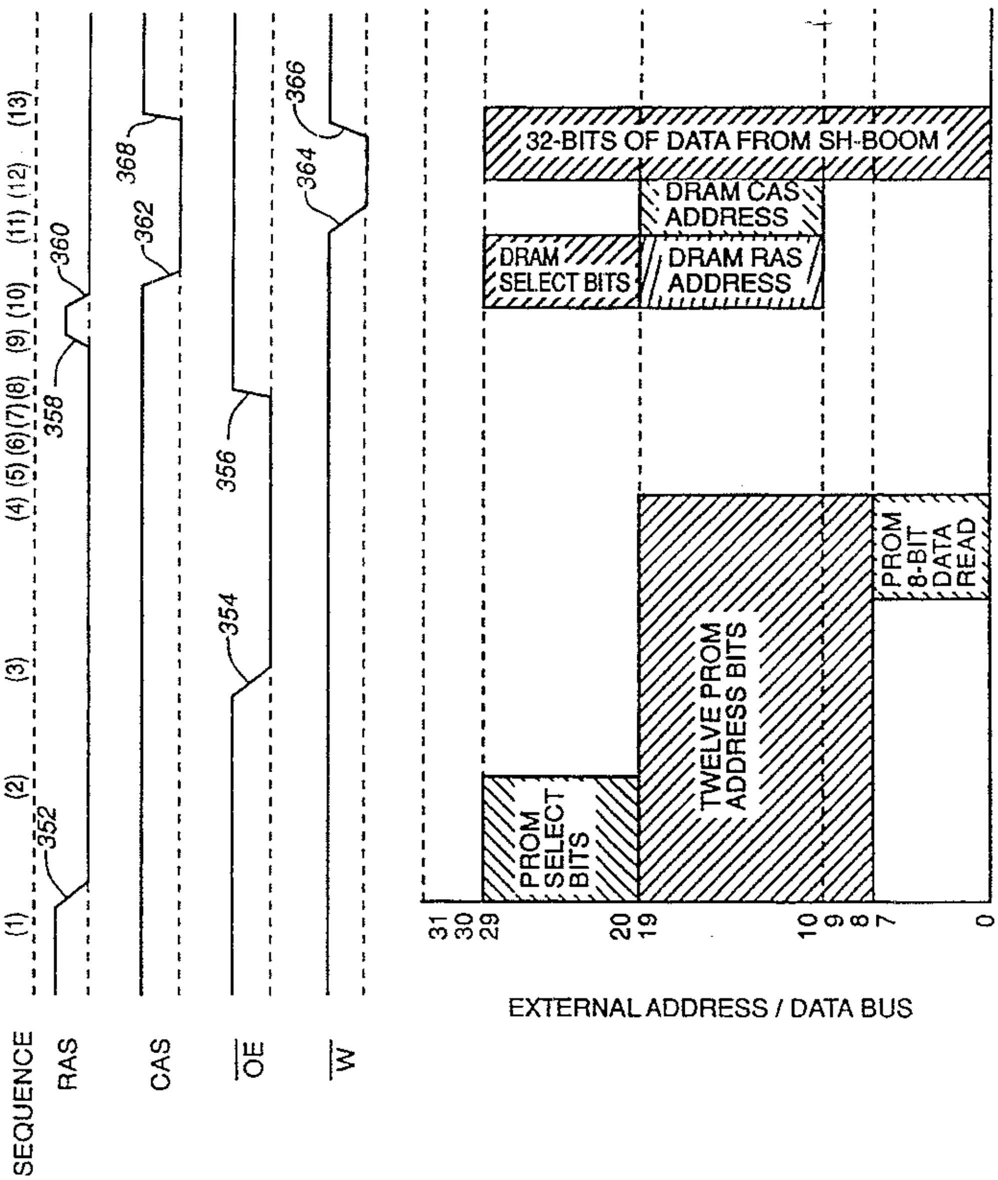

INTERNAL ADDRESS/DATA BUS
118
3
0
EXTERNAL ADDRESS/DATA BUS
HIGH ORDER ADDRESS BITS
RAS ADDRESS BITS
CAS ADDRESS BITS
BYTE SELECT BITS
350
10 BITS
10 BITS
8 BITS
1
1
1
1
RAM SELECT
PROM SELECT
MEMORY/IO DECODE
RAS
10
12
10
8
EPROM
CS
OE
A1
A0
260
10
8
PROM SEL
DRAM
150
A9
A0
OE
W
CAS
PAS
32
RAM SELECT
CAS
W
OE

FIG._10



SEQUENCE
RAS
CAS
OE
W
(1)
(2)
(3)
(4) (5) (6) (7) (8)
(9) (10)
(11) (12)
(13)
352
354
356
358
360
362
364
366
368
PROM SELECT BITS
TWELVE PROM ADDRESS BITS
PROM 8-BIT DATA READ
DRAM SELECT BITS
DRAM RAS ADDRESS
DRAM CAS ADDRESS
32-BITS OF DATA FROM SH-BOOM
31
30
29
20
19
10
9
8
7
0
EXTERNAL ADDRESS / DATA BUS

FIG._11

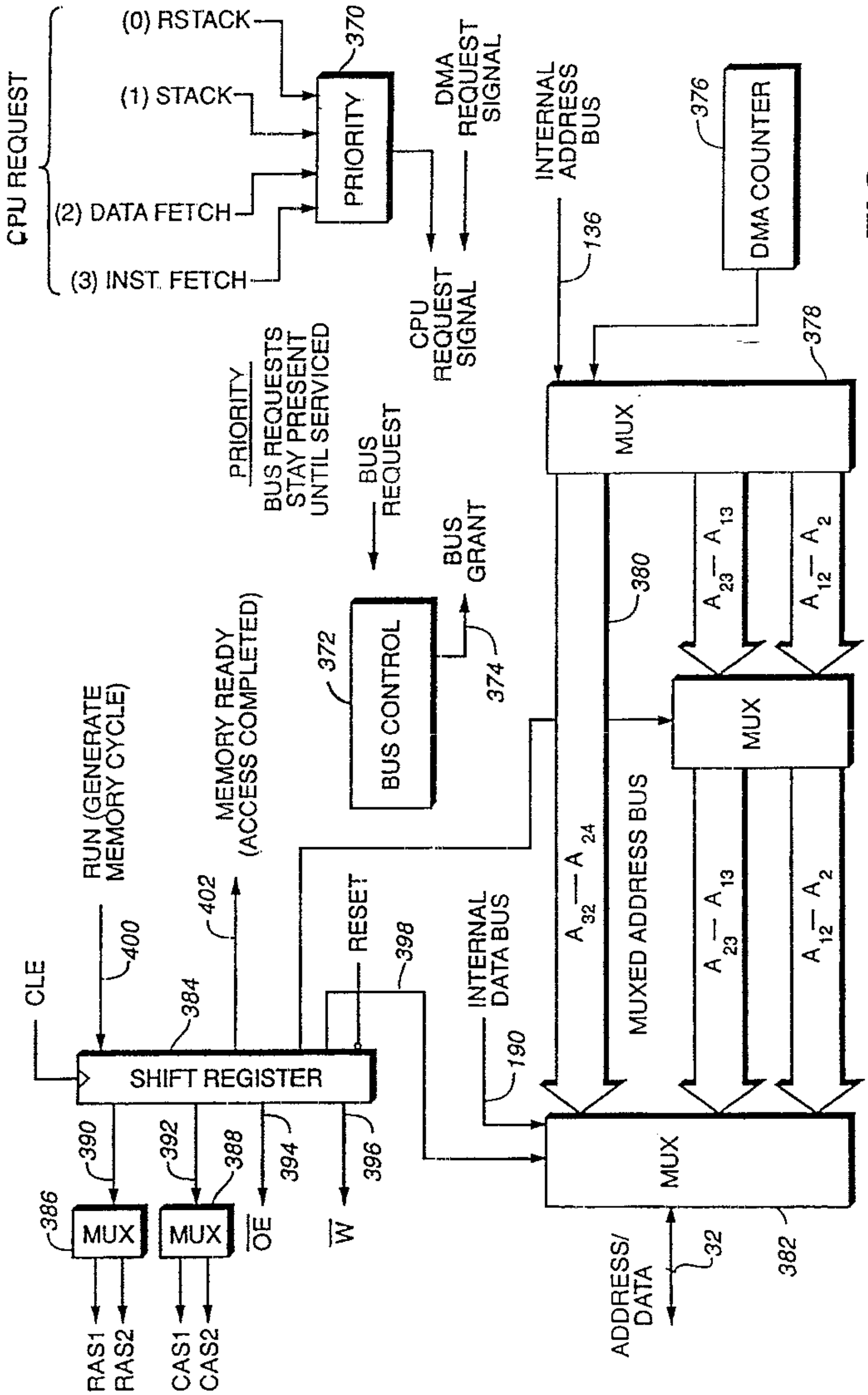

CPU REQUEST
(0) RSTACK
(1) STACK
(2) DATA FETCH
(3) INST. FETCH
PRIORITY
370
DMA REQUEST SIGNAL
CPU REQUEST SIGNAL
PRIORITY
BUS REQUESTS STAY PRESENT UNTIL SERVICED
BUS REQUEST
BUS GRANT
BUS CONTROL
372
374
INTERNAL ADDRESS BUS
136
376
DMA COUNTER
378
MUX
380
A23 — A13
A12 — A2
MUX
RUN (GENERATE MEMORY CYCLE)
MEMORY READY (ACCESS COMPLETED)
CLE
400
402
384
SHIFT REGISTER
RESET
398
INTERNAL DATA BUS
190
A32 — A24
MUXED ADDRESS BUS
A23 — A13
A12 — A2
MUX
386
MUX
390
392
388
MUX
394
396
OE
W
RAS1
RAS2
CAS1
CAS2
ADDRESS/ DATA
32
382

FIG._12

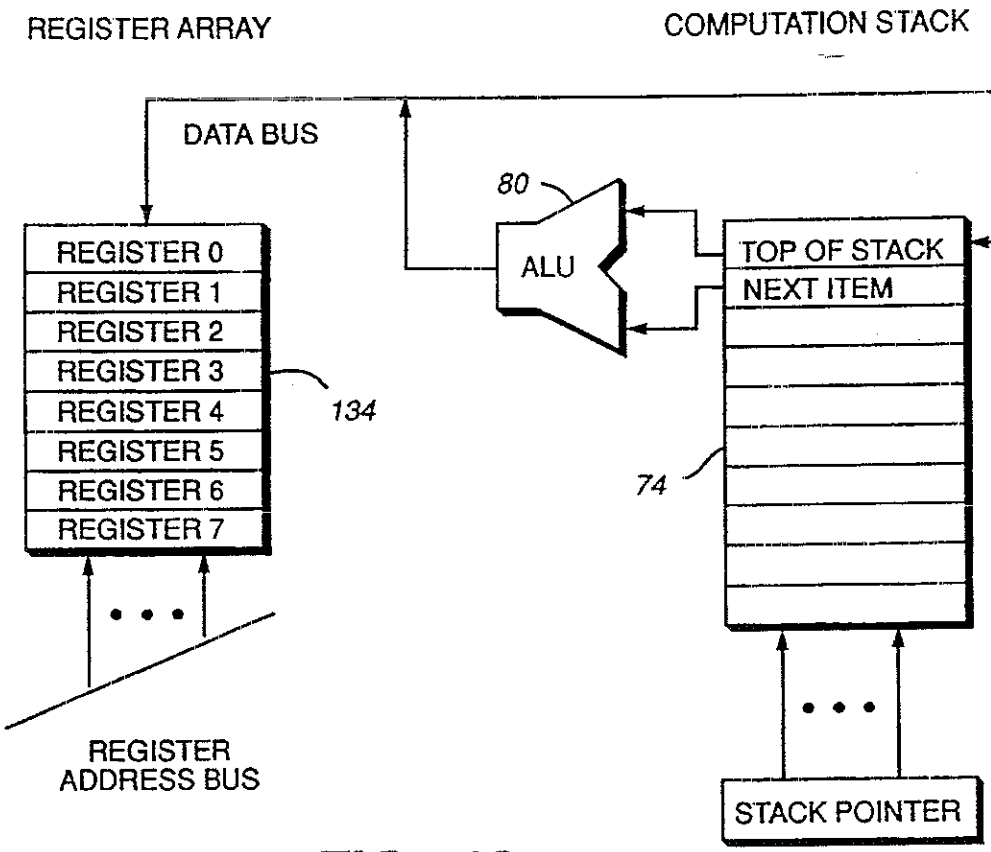

REGISTER ARRAY
COMPUTATION STACK
DATA BUS
80
ALU
TOP OF STACK
NEXT ITEM
REGISTER 0
REGISTER 1
REGISTER 2
REGISTER 3
REGISTER 4
REGISTER 5
REGISTER 6
REGISTER 7
134
74
REGISTER ADDRESS BUS
STACK POINTER

FIG._13

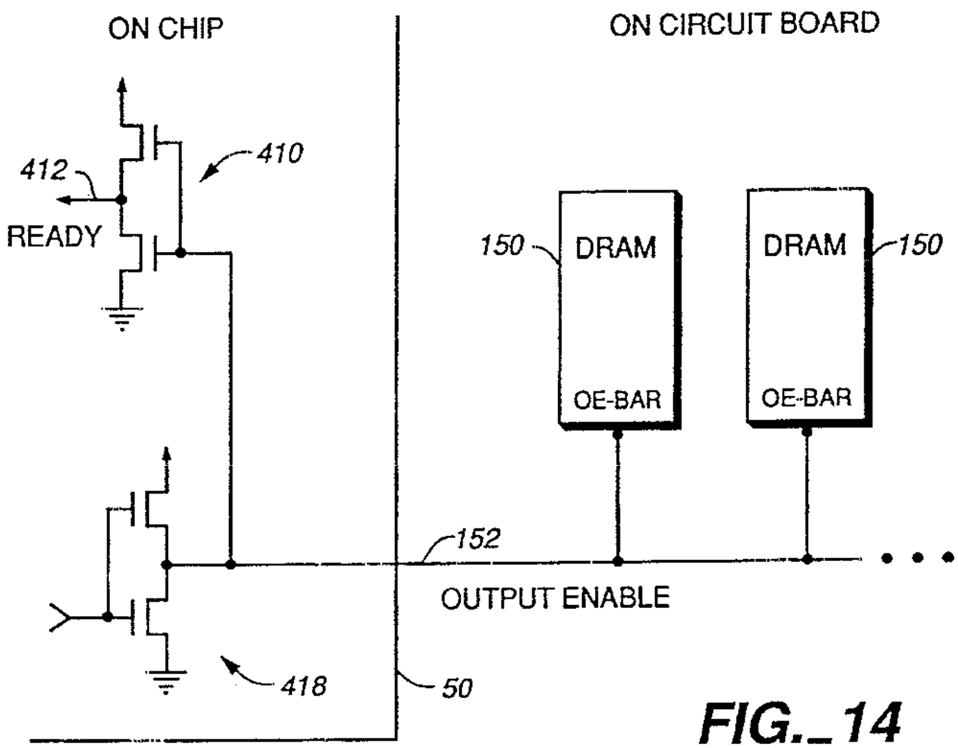

ON CHIP
ON CIRCUIT BOARD
412
410
READY
150
DRAM
DRAM
150
OE-BAR
OE-BAR
152
OUTPUT ENABLE
418
50

FIG._14

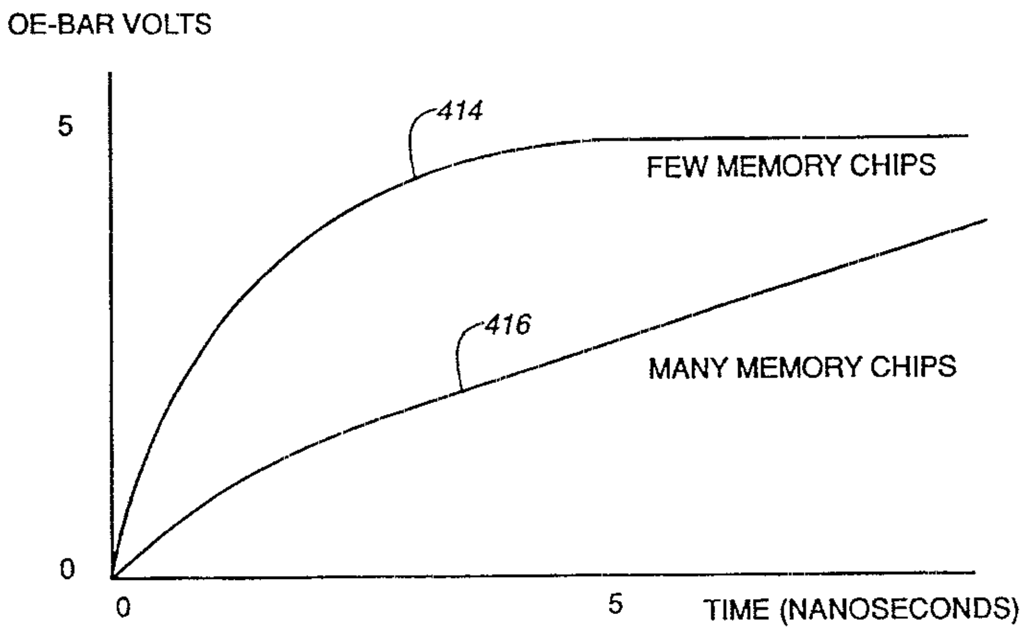

OE-BAR VOLTS
5
414
FEW MEMORY CHIPS
416
MANY MEMORY CHIPS
0
0
5
TIME (NANOSECONDS)

FIG._15

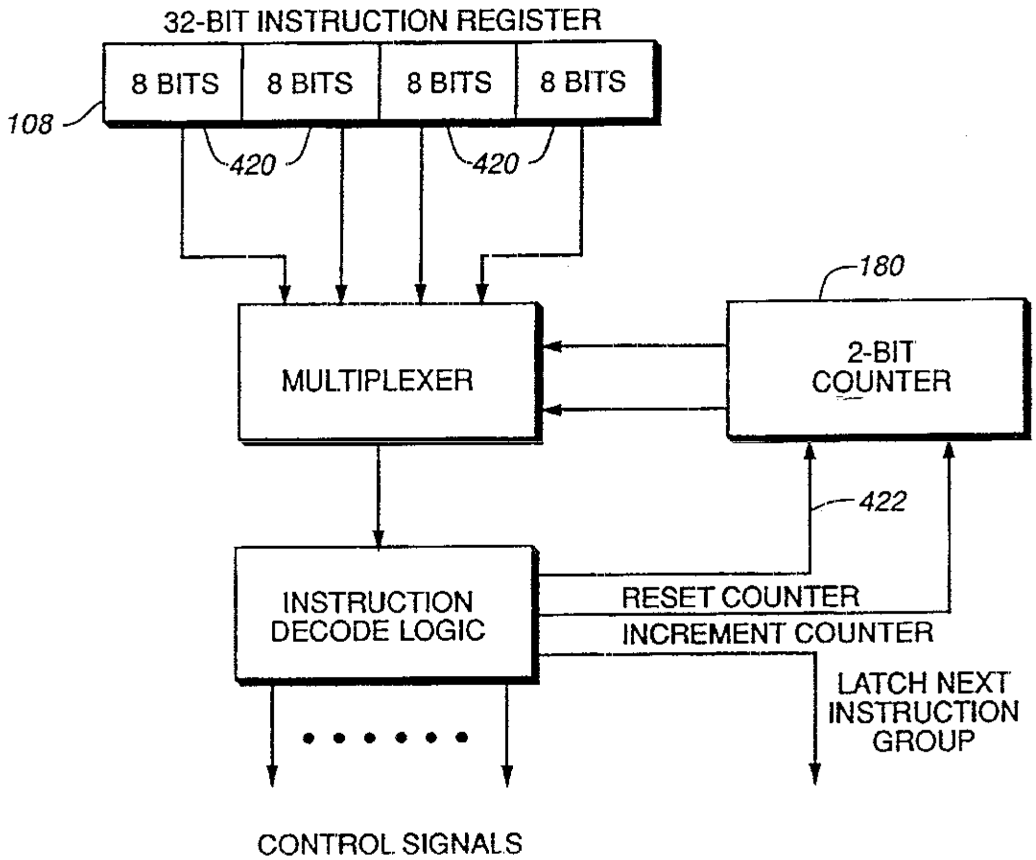

32-BIT INSTRUCTION REGISTER
8 BITS
8 BITS
8 BITS
8 BITS
108
420
420
180
MULTIPLEXER
2-BIT COUNTER
422
INSTRUCTION DECODE LOGIC
RESET COUNTER
INCREMENT COUNTER
LATCH NEXT INSTRUCTION GROUP
CONTROL SIGNALS

FIG._16

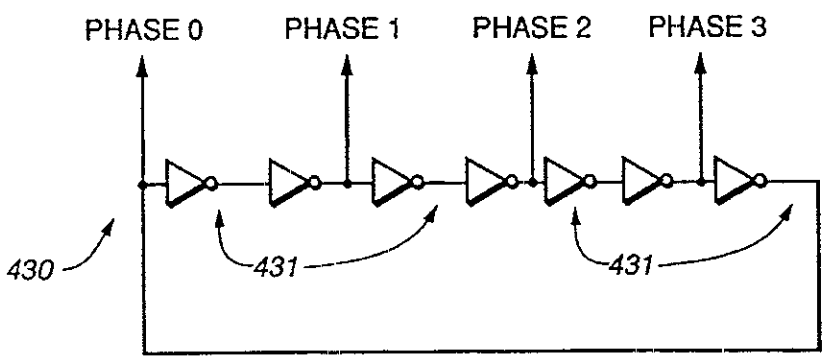

PHASE 0
PHASE 1
PHASE 2
PHASE 3
430
431
431

FIG._18

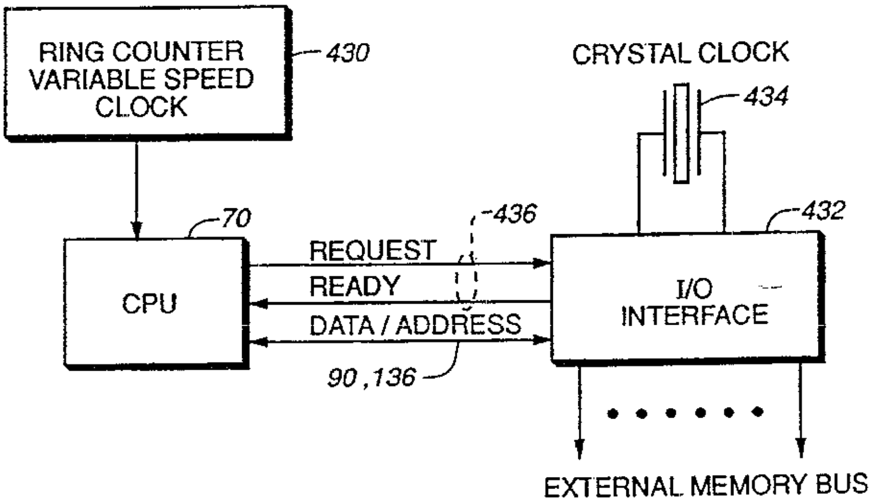

RING COUNTER
VARIABLE SPEED
CLOCK
430
CRYSTAL CLOCK
434
70
436
432
CPU
REQUEST
READY
DATA / ADDRESS
I/O
INTERFACE
90 ,136
EXTERNAL MEMORY BUS

FIG._17

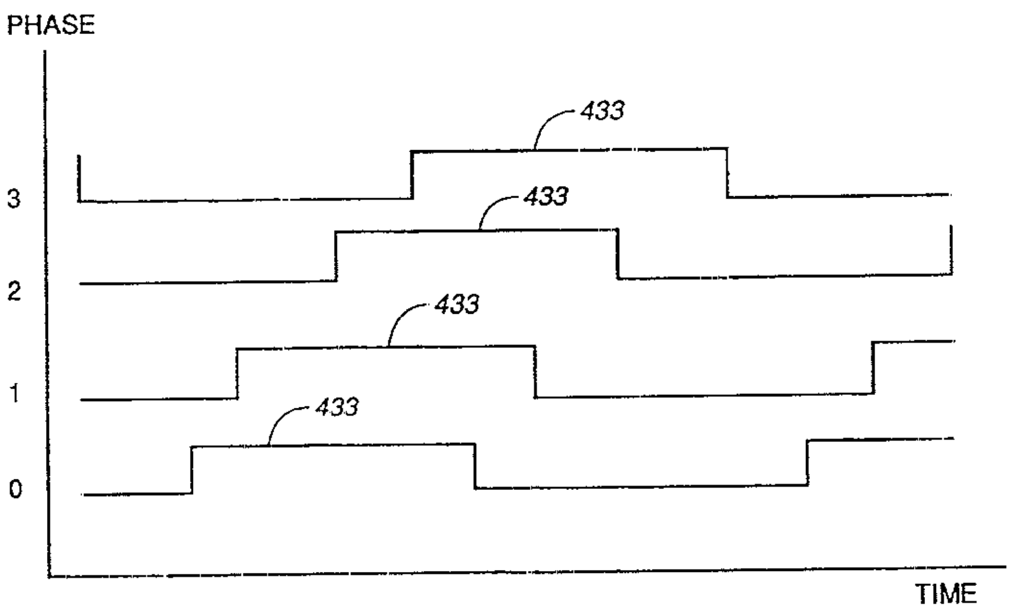
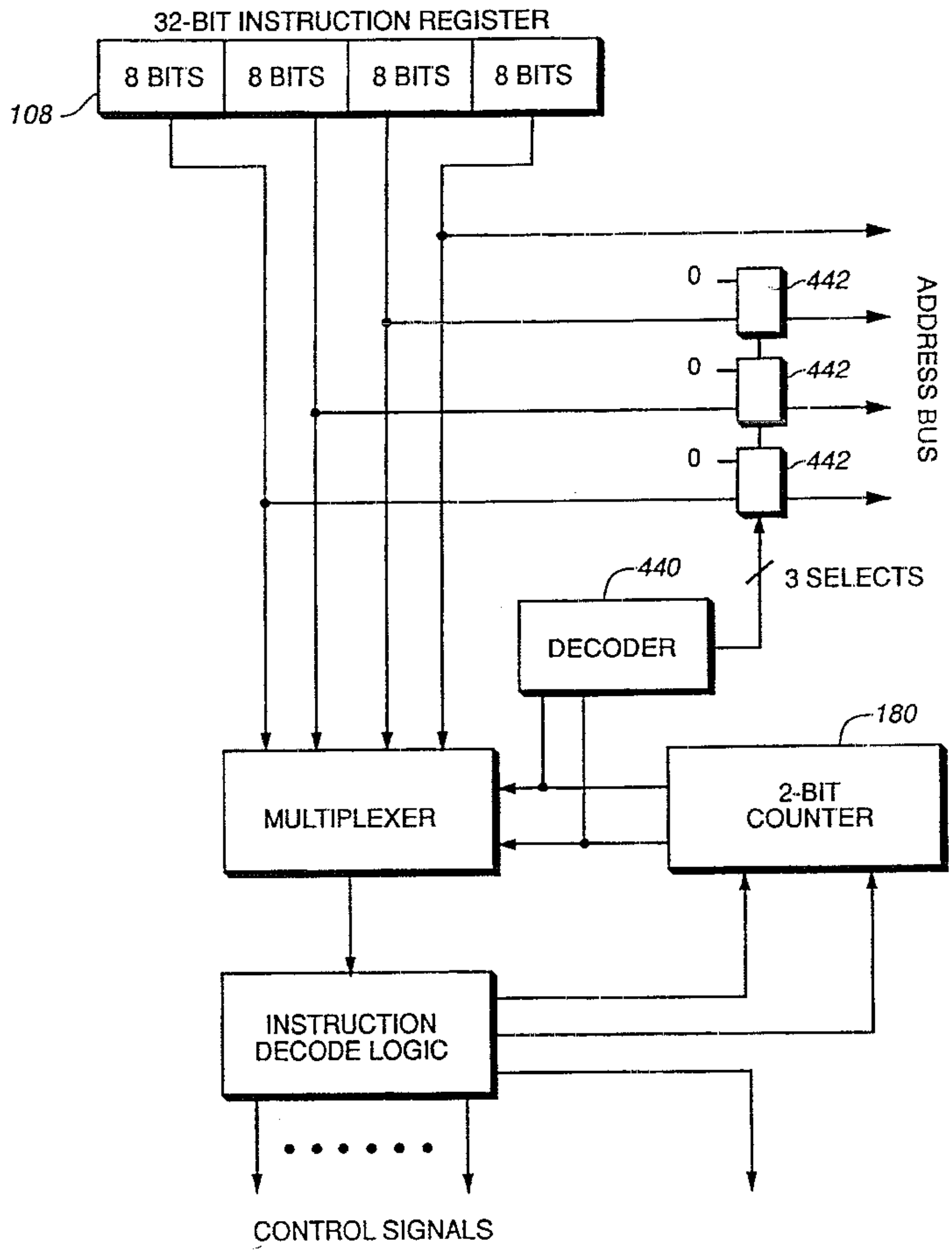

PHASE
433
433
433
433
3
2
1
0
TIME

FIG._19



32-BIT INSTRUCTION REGISTER
8 BITS
8 BITS
8 BITS
8 BITS
108
0
442
0
442
0
442
ADDRESS BUS
440
3 SELECTS
DECODER
180
MULTIPLEXER
2-BIT
COUNTER
INSTRUCTION
DECODE LOGIC
CONTROL SIGNALS

FIG._20

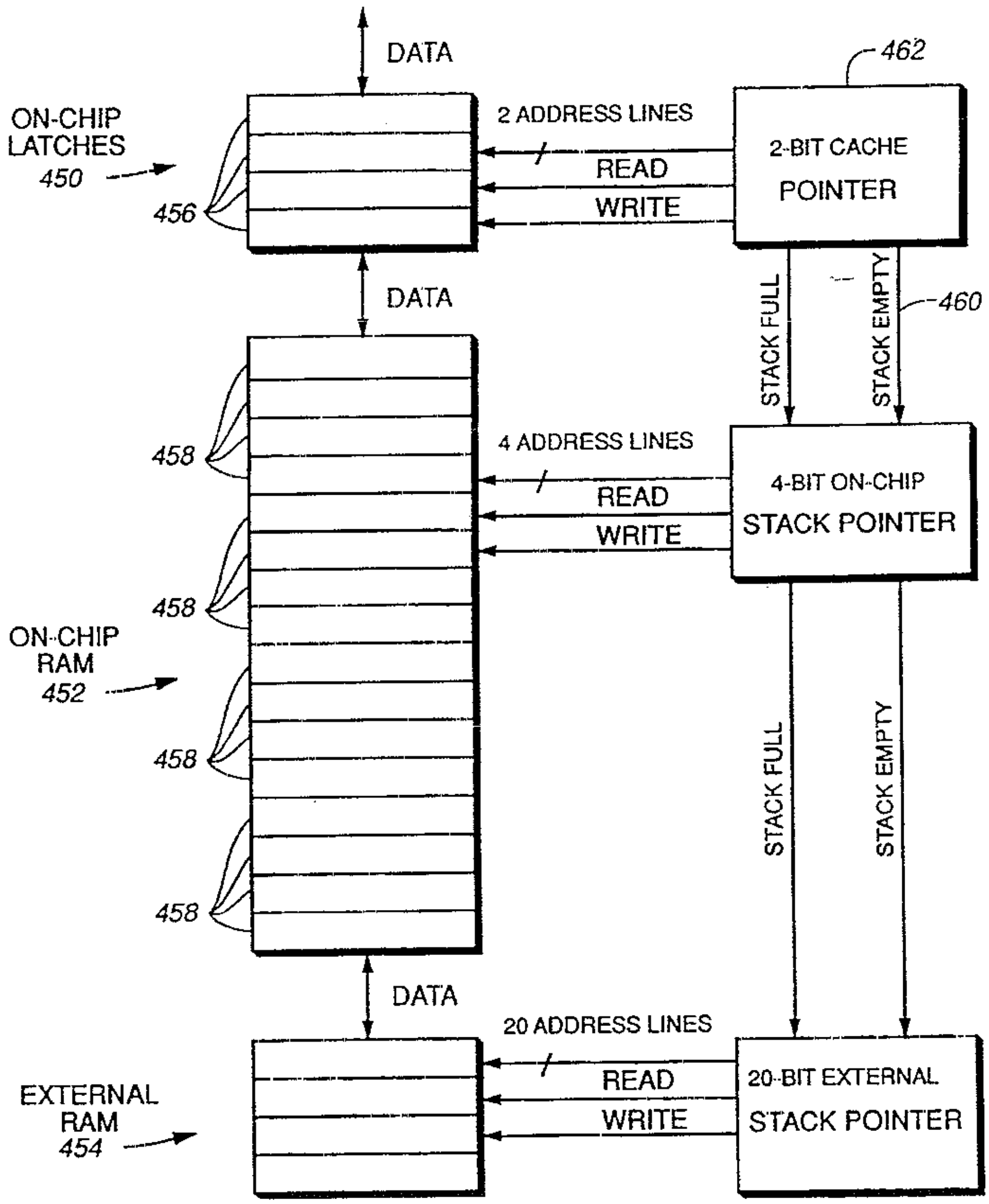

DATA
462
ON-CHIP LATCHES 450
456
2 ADDRESS LINES
READ
WRITE
2-BIT CACHE POINTER
DATA
STACK FULL
STACK EMPTY
460
458
4 ADDRESS LINES
READ
WRITE
4-BIT ON-CHIP STACK POINTER
458
ON-CHIP RAM 452
458
STACK FULL
STACK EMPTY
458
DATA
20 ADDRESS LINES
READ
WRITE
EXTERNAL RAM 454
20-BIT EXTERNAL STACK POINTER

FIG._21

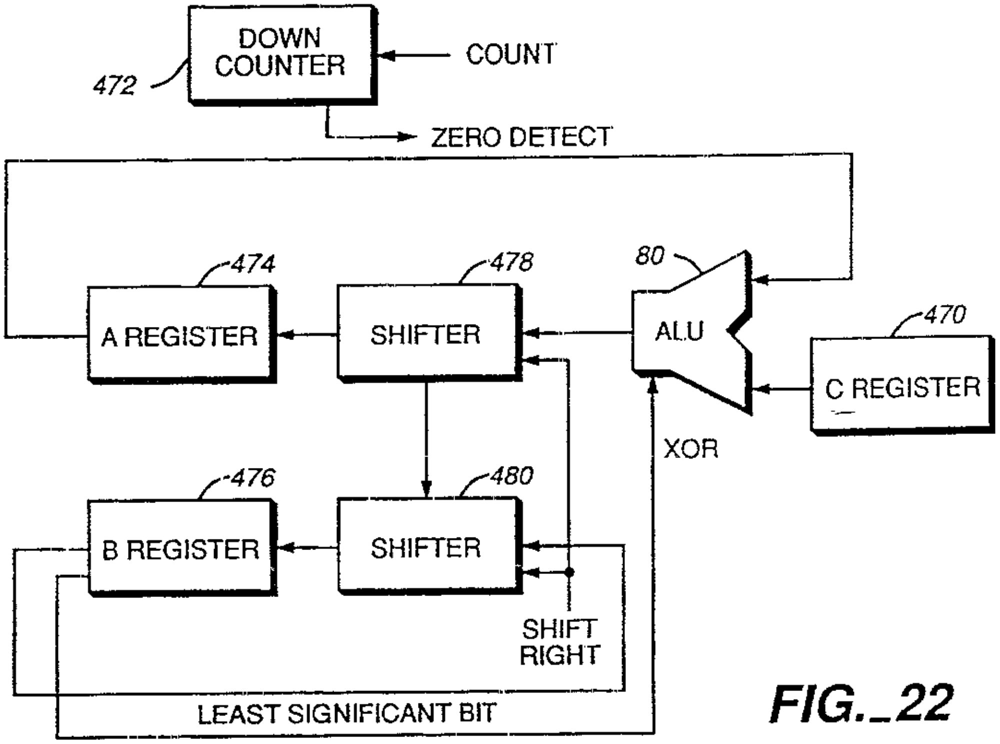

DOWN COUNTER
COUNT
472
ZERO DETECT
474
478
80
A REGISTER
SHIFTER
ALU
470
C REGISTER
XOR
476
480
B REGISTER
SHIFTER
SHIFT RIGHT
LEAST SIGNIFICANT BIT

FIG._22

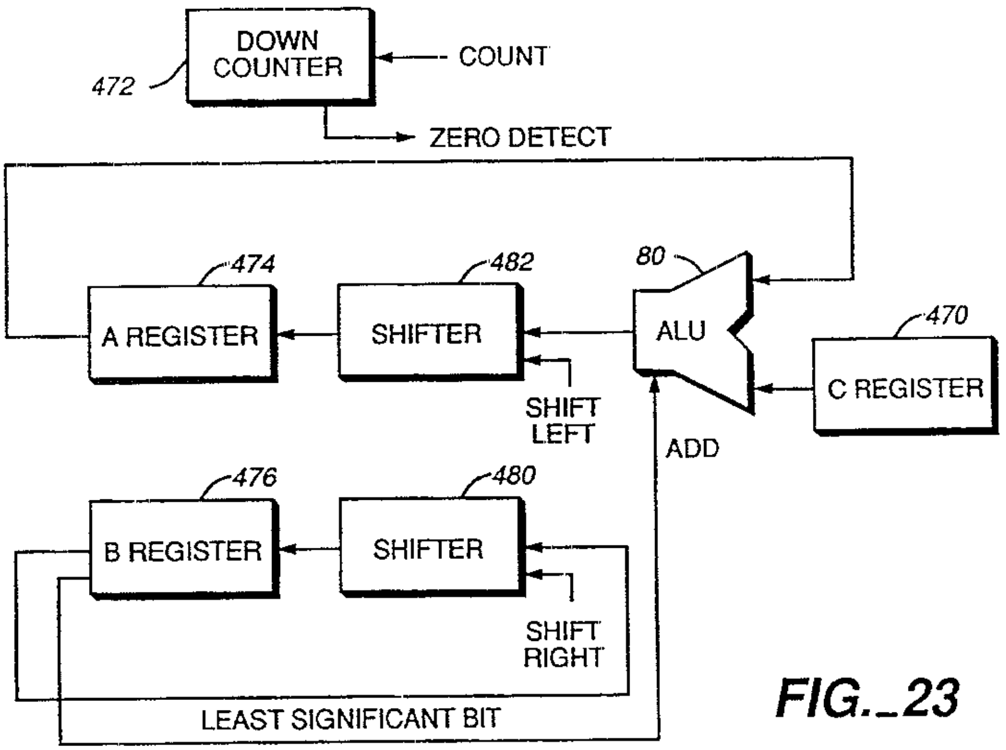

DOWN COUNTER
COUNT
472
ZERO DETECT
474
482
80
A REGISTER
SHIFTER
ALU
470
C REGISTER
SHIFT LEFT
ADD
476
480
B REGISTER
SHIFTER
SHIFT RIGHT
LEAST SIGNIFICANT BIT

FIG._23

HIGH PERFORMANCE, LOW COSI MICROPROCESSOR ARCHITECTURE

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to a simplified, reduced instruction set computer (RISC) microprocessor. More particularly, it relates to such a microprocessor which is capable of performance levels of, for example, 20 million instructions per second (MIPS) at a price of, for example, 20 dollars.

2. Description of the Prior Art

Since the invention of the microprocessor, improvements in its design have taken two different approaches. In the first approach, a brute force gain in performance has been achieved through the provision of greater numbers of faster transistors in the microprocessor integrated circuit and an instruction set of increased complexity. This approach is exemplified by the Motorola 68000 and Intel 80X86 microprocessor families. The trend in this approach is to larger die sizes and packages, with hundreds of pinouts.

More recently, it has been perceived that performance gains can be achieved through comparative simplicity both in the microprocessor integrated circuit itself and in its instruction set. This second approach provides RISC microprocessors, and is exemplified by the Sun SPARC and The Intel 8960 microprocessors. However, even with this approach as conventionally practiced, the packages for the microprocessor are large, in order to accommodate the large number of pinouts that continue to be employed. A need therefore remains for further simplification of high performance microprocessors

With conventional high performance microprocessors, fast static memories are required for direct connection to the microprocessors in order to allow memory accesses that are fast enough to keep up with the microprocessors Slower dynamic random access memories (DRAMs) are used with such microprocessors only in a hierarchical memory arrangement, with the static memories acting as a buffer between the microprocessors and the DRAMs. The necessity to use static memories increases cost of the resulting systems.

Conventional microprocessors provide direct memory accesses (DMA) for system peripheral units through DMA controllers, which may be located on the microprocessor integrated circuit, or provided separately. Such DMA controllers can provide routine handling of DMA requests and responses, but some processing by the main central processing unit (CPU) of the microprocessor is required

SUMMARY OF THE INVENTION

Accordingly, it is an object of this invention to provide a microprocessor with a reduced pin count and cost compared to conventional microprocessors.

It is another object of the invention to provide a high performance microprocessor that can be directly connected to DRAMs without sacrificing microprocessor speed.

It is a further object of the invention to provide a high performance microprocessor in which DMA does not require use of the main CPU during DMA requests and responses and which provides very rapid DMA response with predictable response times.

The attainment of these and related objects may be achieved through use of the novel high performance, low cost microprocessor herein disclosed. In accordance with one aspect of the invention, a microprocessor system in accordance with this invention has a central processing unit, a dynamic random access memory and a bus connecting the central processing unit to the dynamic random access memory. There is a multiplexing means on the bus between the central processing unit and the dynamic random access memory. The multiplexing means is connected and configured to provide row addresses, column addresses and data on the bus.

In accordance with another aspect of the invention, the microprocessor system has a means connected to the bus for fetching instructions for the central processing unit on the bus. The means for fetching instructions is configured to fetch multiple sequential instructions in a single memory cycle. In a variation of this aspect of the invention, a programmable read only memory containing instructions for the central processing unit is connected to the bus. The means for fetching instructions includes means for assembling a plurality of instructions from the programmable read only memory and storing the plurality of instructions in the dynamic random access memory.

In another aspect of the invention. the microprocessor system includes a central processing unit. a direct memory access processing unit and a memory connected by a bus The direct memory access processing unit includes means for fetching instructions for the central processing unit and for fetching instructions for the direct memory access processing unit on the bus.

In a further aspect of the invention, the microprocessor system, including the memory, is contained in an integrated circuit The memory is a dynamic random access memory, and the means for fetching multiple instructions includes a column latch for receiving the multiple instructions

In still another aspect of the invention, the microprocessor system additionally includes an instruction register for the multiple instructions connected to the means for fetching instructions. A means is connected to the instruction register for supplying the multiple instructions in succession from the instruction register. A counter is connected to control the means for supplying the multiple instructions to supply the multiple instructions in succession A means for decoding the multiple instructions is connected to receive the multiple instructions in succession from the means for supplying the multiple instructions. The counter is connected to said means for decoding to receive incrementing and reset control signals from the means for decoding The means for decoding is configured to supply the reset control signal to the counter and to supply a control signal to the means for fetching instructions in response to a SKIP instruction in the multiple instructions. In a modification of this aspect of the invention, the microprocessor system additionally has a loop counter connected to receive a decrement control signal from the means for decoding. The means for decoding is configured to supply the reset control signal to the counter and the decrement control signal to the loop counter in response to a MICROLOOP instruction in the multiple instructions In a further modification to this aspect of the invention, the means for decoding is configured to control the counter in response to an instruction utilizing a variable width operand A means is connected to

the counter to select the variable width operand in response to the counter.

In a still further aspect of the invention, the microprocessor system includes an arithmetic logic unit. A first push down stack is connected to the arithmetic logic unit. The first push down stack includes means for storing a top item connected to a first input of the arithmetic logic unit and means for storing a next item connected to a second input of the arithmetic logic unit The arithmetic logic unit has an output connected to the means for storing a top item. The means for storing a top item is connected to provide an input to a register file The register file desirably is a second push down stack, and the means for storing a top item and the register file are bidirectionally connected.

In another aspect of the invention, a data processing system has a microprocessor including a sensing circuit and a driver circuit, a memory, and an output enable line connected between the memory, the sensing circuit and the driver circuit. The sensing circuit is configured to provide a ready signal when the output enable line reaches a predetermined electrical level, such as a voltage. The microprocessor is configured so that the driver circuit provides an enabling signal on the output enable line responsive to the ready signal.

In a further aspect of the invention, the microprocessor system has a ring counter variable speed system clock connected to the central processing unit. The central processing unit and the ring counter variable speed system clock are provided in a single integrated circuit. An input/output interface is connected to exchange coupling control signals, addresses and data with the input/output interface. A second clock independent of the ring counter variable speed system clock is connected to the input/output interface.

In yet another aspect of the invention, a push down stack is connected to the arithmetic logic unit. The push down stack includes means for storing a top item connected to a first input of the arithmetic logic unit and means for storing a next item connected to a second input of the arithmetic logic unit. The arithmetic logic unit has an output connected to the means for storing a top item. The push down stack has a first plurality of stack elements configured as latches and a second plurality of stack elements configured as a random access memory. The first and second plurality of stack elements and the central processing unit are provided in a single integrated circuit. A third plurality of stack elements is configured as a random access memory external to the single integrated circuit In this aspect of the invention, desirably a first pointer is connected to the first plurality of stack elements, a second pointer connected to the second plurality of stack elements, and a third pointer is connected to the third plurality of stack elements The central processing unit is connected to pop items from the first plurality of stack elements The first stack pointer is connected to the second stack pointer to pop a first plurality of items from the second plurality of stack elements when the first plurality of stack elements are empty from successive pop operations by the central processing unit. The second stack pointer is connected to the third stack pointer to pop a second plurality of items from the third plurality of stack elements when the second plurality of stack elements are empty from successive pop operations by the central processing unit.

In another aspect of the invention, a first register is connected to supply a first input to the arithmetic logic unit A first shifter is connected between an output of the arithmetic logic unit and the first register. A second register is connected to receive a starting polynomial value. An output of the second register is connected to a second shifter A least significant bit of the second register is connected to The arithmetic logic unit. A third register is connected to supply feedback terms of a polynomial to the arithmetic logic unit. A down counter, for counting down a number corresponding to digits of a polynomial to be generated, is connected to the arithmetic logic unit. The arithmetic logic unit is responsive to a polynomial instruction to carry out an exclusive OR of the contents of the first register with the contents of the third register if the least significant bit of the second register is a "ONE"and to pass the contents of the first register unaltered if the least significant bit of the second register is a "ZERO", until the down counter completes a count. The polynomial to be generated results in said first register.

In still another aspect of the invention, a result register is connected to supply a first input to the arithmetic logic unit. A first, left shifting shifter is connected between an output of the arithmetic logic unit and the result register. A multiplier register is connected to receive a multiplier in bit reversed form. An output of the multiplier register is connected to a second, right shifting shifter A least significant bit of the multiplier register is connected to the arithmetic logic unit. A third register is connected to supply a multiplicand to said arithmetic logic unit. A down counter, for counting down a number corresponding to one less than the number of digits of the multiplier, is connected to the arithmetic logic unit The arithmetic logic unit is responsive to a multiply instruction to add the contents of the result register with the contents of the third register, when the least significant bit of the multiplier register is a "ONE" and to pass the contents of the result register unaltered, until the down counter completes a count The product results in the result register

The attainment of the foregoing and related objects, advantages and features of the invention should be more readily apparent to those skilled in the art, after review of the following more detailed description of the invention, taken together with the drawings, in which:

BRIEF DESCRIPTION OF THE DRAWINGS

FIG 1 is an external, plan view of an integrated circuit package incorporating a microprocessor in accordance with the invention

FIG. 2 is a block diagram of a microprocessor in accordance with the invention.

FIG 3 is a block diagram of a portion of a data processing system incorporating the microprocessor of FIGS. 1 and 2

FIG. 4 is a more detailed block diagram of a portion of the microprocessor shown in FIG 2

FIG 5 is a more detailed block diagram of another portion of the microprocessor shown in FIG. 2.

FIG 6 is a block diagram of another portion of the data processing system shown in part in FIG 3 and incorporating the microprocessor of FIGS. 1–2 and 4–5.

FIGS. 7 and 8 are layout diagrams for the data processing system shown in part in FIGS 3 and 6

FIG 9 is a layout diagram of a second embodiment of a microprocessor in accordance with the invention in a data processing system on a single integrated circuit

FIG. 10 is a more detailed block diagram of a portion of the data processing system of FIGS 7 and 8.

FIG 11 is a timing diagram useful for understanding operation of the system portion shown in FIG. 12.

FIG 12 is another more detailed block diagram of a further portion of the data processing system of FIGS 7 and 8

FIG 13 is a more detailed block diagram of a portion of the microprocessor shown in FIG. 2.

FIG 14 is a more detailed block and schematic diagram of a portion of the system shown in FIGS 3 and 7–8.

FIG 15 is a graph useful for understanding operation of the system portion shown in FIG. 14.

FIG. 16 is a more detailed block diagram showing part of the system portion shown in FIG 4.

FIG. 17 is a more detailed block diagram of a portion of the microprocessor shown in FIG. 2

FIG. 18 is a more detailed block diagram of part of the microprocessor portion shown in FIG. 17

FIG. 19 is a set of waveform diagrams useful for understanding operation of the part of the microprocessor portion shown in FIG. 18

FIG. 20 is a more detailed block diagram showing another part of the system portion shown in FIG. 4.

FIG. 21 is a more detailed block diagram showing another part of the system portion shown in FIG 4

FIGS. 22 and 23 are more detailed block diagrams showing another part of the system portion shown in FIG 4

DETAILED DESCRIPTION OF THE INVENTION

Overview

The microprocessor of this invention is desirably implemented as a 32-bit microprocessor optimized for:

HIGH EXECUTION SPEED, and

LOW SYSTEM COST.

In this embodiment, the microprocessor can be thought of as 20 MIPS for 20 dollars. Important distinguishing features of the microprocessor are:

Uses low-cost commodity DYNAMIC RAMS to run 20 MIPS

4 instruction fetch per memory cycle

On-chip fast page-mode memory management

Runs fast without external cache

Requires few interfacing chips

Crams 32-bit CPU in 44 pin SOJ package

The instruction set is organized so that most operations can be specified with 8-bit instructions. Two positive products of this philosophy are:

Programs are smaller,

Programs can execute much faster

The bottleneck in most computer systems is the memory bus. The bus is used to fetch instructions and fetch and store data. The ability to fetch four instructions in a single memory bus cycle significantly increases the bus availability to handle data.

Turning now to the drawings, more particularly to FIG. 1, there is shown a packaged 32-bit microprocessor 50 in a 44-pin plastic leadless chip carrier, shown approximately 100 times its actual size of about 0.8 inch on a side. The fact that the microprocessor 50 is provided as a 44-pin package represents a substantial departure from typical microprocessor packages, which usually have about 200 input/output (I/O) pins. The microprocessor 50 is rated at 20 million instructions per second (MIPS). Address and data lines 52, also labelled D0–D31, are shared for addresses and data without speed penalty as a result of the manner in which the microprocessor 50 operates, as will be explained below

Dynamic Ram

In addition to the low cost 44-pin package, another unusual aspect of the high performance microprocessor 50 is that it operates directly with dynamic random access memories (DRAMs), as shown by row address strobe (RAS) and column address strobe (CAS) I/O pins 54. The other I/O pins for the microprocessor 50 include VDD pins 56, VSS pins 58, output enable pin 60, write pin 62, clock pin 64 and reset pin 66

All high speed computers require high speed and expensive memory to keep up. The highest speed static RAM memories cost as much as ten times as much as slower dynamic RAMs. This microprocessor has been optimized to use low-cost dynamic RAM in high-speed page-mode. Page-mode dynamic RAMs offer static RAM performance without the cost penalty. For example, low-cost 85 nsec. dynamic RAMs access at 25 nsec when operated in fast page-mode. Integrated fast page-mode control on the microprocessor chip simplifies system interfacing and results in a faster system.

Details of the microprocessor 50 are shown in FIG. 2. The microprocessor 50 includes a main central processing unit (CPU) 70 and a separate direct memory access (DMA) CPU 72 in a single integrated circuit making up the microprocessor 50. The main CPU 70 has a first 16 deep push down stack 74, which has a top item register 76 and a next item register 78, respectively connected to provide inputs to an arithmetic logic unit (ALU) 80 by lines 82 and 84. An output of the ALU 80 is connected to the top item register 76 by line 86. The output of the top item register at 82 is also connected by line 88 to an internal data bus 90.

A loop counter 92 is connected to a decrementer 94 by lines 96 and 98. The loop counter 92 is bidirectionally connected to the internal data bus 90 by line 100. Stack pointer 102, return stack pointer 104, mode register 106 and instruction register 108 are also connected to the internal data bus 90 by lines 110, 112, 114 and 116, respectively. The internal data bus 90 is connected to memory controller 118 and to gate 120. The gate 120 provides inputs on lines 122, 124, and 126 to X register 128, program counter 130 and Y register 132 of return push down stack 134. The X register 128, program counter 130 and Y register 132 provide outputs to internal address bus 136 on lines 138, 140 and 142. The internal address bus provides inputs to the memory controller 118 and to an incrementer 144. The incrementer 144 provides inputs to the X register, program counter and Y register via lines 146, 122, 124 and 126. The DMA CPU 72 provides inputs to the memory controller 118 on line 148. The memory controller 118 is connected to a RAM (not shown) by address/data bus 151 and control lines 153.

FIG. 2 shows that the microprocessor 50 has a simple architecture. Prior art RISC microprocessors are substantially more complex in design. For example, the SPARC RISC microprocessor has three times the gates of the microprocessor 50, and the Intel 8960 RISC microprocessor has 20 times the gates of the microprocessor 50. The speed of this microprocessor is in substantial part due to this simplicity. The architecture incorporates push down stacks and register write to achieve this simplicity.

The microprocessor 50 incorporates an I/O that has been tuned to make heavy use of resources provided on

the integrated circuit chip. On chip latches allow use of the same I/O circuits to handle three different things: column addressing, row addressing and data, with a slight to non-existent speed penalty. This triple bus multiplexing results in fewer buffers to expand, fewer interconnection lines, fewer I/O pins and fewer internal buffers

The provision of on-chip DRAM control gives a performance equal to that obtained with the use of static RAMs. As a result, memory is provided at ¼ the system cost of static RAM used in most RISC systems

The microprocessor **50** fetches 4 instructions per memory cycle; the instructions are in an 8-bit format, and this is a 32-bit microprocessor System speed is therefore 4 times the memory bus bandwidth This ability enables the microprocessor to break the Von Neumann bottleneck of the speed of getting the next instruction. This mode of operation is possible because of the use of a push down stack and register array The push down stack allows the use of implied addresses, rather than the prior art technique of explicit addresses for two sources and a destination

Most instructions execute in 20 nanoseconds in the microprocessor **50** The microprocessor can therefore execute instructions at **50** peak MIPS without pipeline delays This is a function of the small number of gates in the microprocessor 50 and the high degree of parallelism in the architecture of the microprocessor.

FIG. 3 shows how column and row addresses are multiplexed on lines D8–D14 of the microprocessor **50** for addressing DRAM 150 from I/O pins **52**. The DRAM 150 is one of eight, but only one DRAM 150 has been shown for clarity As shown, the lines D11–D18 are respectively connected to row address inputs A0–A8 of the DRAM **150**. Additionally, lines D12–D15 are connected to the data inputs DQ1–DQ4 of the DRAM **15** The output enable, write and column address strobe pins **54** are respectively connected to the output enable, write and column address strobe inputs of the DRAM 150 by lines **152** The row address strobe pin **54** is connected through row address strobe decode logic **154** to the row address strobe input of the DRAM **150** by lines **156** and **158**.

D0–D7 pins **52** (FIG. **1**) are idle when the microprocessor **50** is outputting multiplexed row and column addresses on D11–D18 pins **52** The D0–D7 pins **52** can therefore simultaneously be used for I/O when right justified I/O is desired. Simultaneous addressing and I/O can therefore be carried out

FIG. **4** shows how the microprocessor **50** is able to achieve performance equal to the use of static RAMS with DRAMs through multiple instruction fetch in a single clock cycle and instruction fetch-ahead Instruction register **108** receives four 8-bit byte instruction words **1–4** on 32-bit internal data bus **90**. The four instruction byte **1–4** locations of the instruction register **108** are connected to multiplexer **170** by busses **172, 174, 176** and **178**, respectively. A microprogram counter **180** is connected to the multiplexer **170** by lines **182** The multiplexer **170** is connected to decoder **184** by bus **186**. The decoder **184** provides internal signals to the rest of the microprocessor **50** on lines **188**

Most significant bits **190** of each instruction byte **14** location are connected to a 4-input decoder **192** by lines **194** The output of decoder **192** is connected to memory controller **118** by line **196** Program counter **130** is connected to memory controller **118** by internal address bus **136**, and the instruction register **108** is connected to the memory controller **118** by the internal data bus **90** Address/data bus **198** and control bus **200** are connected to the DRAMS **150** (FIG 3).

In operation, when the most significant bits **190** of remaining instructions **1–4** are "1" in a clock cycle of the microprocessor **50**, there are no memory reference instructions in the queue The output of decoder **192** on line **196** requests an instruction fetch ahead by memory controller **118** without interference with other accesses. While the current instructions in instruction register **108** are executing, the memory controller **118** obtains the address of the next set of four instructions from program counter **130** and obtains that set of instructions By the time the current set of instructions has completed execution, the next set of instructions is ready for loading into the instruction register

Details of the DMA CPU **72** are provided in FIG. 5 Internal data bus **90** is connected to memory controller **118** and to DMA instruction register **210**. The DMA instruction register **210** is connected to DMA program counter **212** by bus **214**, to transfer size counter **216** by bus **218** and to timed transfer interval counter **220** by bus **222**. The DMA instruction register **210** is also connected to DMA I/O and RAM address register **224** by line **226** The DMA I/O and RAM address register **224** is connected to the memory controller **118** by memory cycle request line **228** and bus **230** The DMA program counter **212** is connected to the internal address bus **136** by bus **232** The transfer size counter **216** is connected to a DMA instruction done decrementer **234** by lines **236** and **238** The decrementer **234** receives a control input on memory cycle acknowledge line **240**. When transfer size counter **216** has completed its count, it provides a control signal to DMA program counter **212** on line **242** Timed transfer interval counter **220** is connected to decrementer **244** by lines **246** and **248** The decrementer **244** receives a control input from a microprocessor system clock on line **250**

The DMA CPU 72 controls itself and has the ability to fetch and execute instructions. It operates as a coprocessor to the main CPU 70 (FIG. 2) for time specific processing.

FIG. **6** shows how the microprocessor **50** is connected to an electrically programmable read only memory (EPROM) **260** by reconfiguring the data lines **52** so that some of the data lines **52** are input lines and some of them are output lines. Data lines **52** D0–D7 provide data to and from corresponding data terminals **262** of the EPROM **260** Data lines **52** D9–D18 provide addresses to address terminals **264** of the EPROM **260**. Data lines **52** D19–D31 provide inputs from the microprocessor **50** to memory and I/O decode logic **266**. RAS **0/1** control line **268** provides a control signal for determining whether the memory and I/O decode logic provides a DRAM RAS output on line **270** or a column enable output for the EPROM **260** on line **272**. Column address strobe terminal **60** of the microprocessor **50** provides an output enable signal on line **274** to the corresponding terminal **276** of the EPROM **260**.

FIGS 7 and **8** show the front and back of a one card data processing system **280** incorporating the microprocessor **50**, MSM514258-10 type DRAMs **150** totalling 2 megabytes, a Motorola 50 MegaHertz crystal oscillator clock **282**, I/O circuits **284** and a 27256 type EPROM **260** The I/O circuits **284** include a 74HC04 type high speed hex inverter circuit **286**, an IDT39C828 type 10-bit inverting buffer circuit **288**, an IDT39C822 type 10-bit inverting register circuit **290**, and two

IDT39C823 type 9-bit non-inverting register circuits 292. The card 280 is completed with a MAX12V type DC-DC converter circuit 294, 34-pin dual AMP type headers 296, a coaxial female power connector 298, and a 3-pin AMP right angle header 300. The card 280 is a low cost, imbeddable product that can be incorporated in larger systems or used as an internal development tool.

The microprocessor 50 is a very high performance (50 MHz) RISC influenced 32-bit CPU designed to work closely with dynamic RAM. Clock for clock, the microprocessor 50 approaches the theoretical performance limits possible with a single CPU configuration. Eventually, the microprocessor 50 and any other processor is limited by the bus bandwidth and the number of bus paths. The critical conduit is between the CPU and memory.

One solution to the bus bandwidth/bus path problem is to integrate a CPU directly onto the memory chips, giving every memory a direct bus to the CPU. FIG. 9 shows another microprocessor 310 that is provided integrally with 1 megabit of DRAM 311 in a single integrated circuit 312. Until the present invention, this solution has not been practical, because most high performance CPUs require from 500,000 to 1,000,000 transistors and enormous die sizes just by themselves. The microprocessor 310 is equivalent to the microprocessor 50 in FIGS. 1–8. The microprocessors 50 and 310 are the most transistor efficient high performance CPUs in existence, requiring fewer than 50,000 transistors for dual processors 70 and 72 (FIG. 2) or 314 and 316 (less memory). The very high speed of the microprocessors 50 and 310 is to a certain extent a function of the small number of active devices. In essence, the less silicon gets in the way, the faster the electrons can get where they are going.

The microprocessor 310 is therefore the only CPU suitable for integration on the memory chip die 312. Some simple modifications to the basic microprocessor 50 to take advantage of the proximity to the DRAM array 311 can also increase the microprocessor 50 clock speed by 50 percent, and probably more.

The microprocessor 310 core on board the DRAM die 312 provides most of the speed and functionality required for a large group of applications from automotive to peripheral control. However, the integrated CPU 310/DRAM 311 concept has the potential to redefine significantly the way multiprocessor solutions can solve a spectrum of very compute intensive problems. The CPU 310/DRAM 311 combination eliminates the Von Neumann bottleneck by distributing it across numerous CPU/DRAM chips 312. The microprocessor 310 is a particularly good core for multiprocessing, since it was designed with the SDI targeting array in mind, and provisions were made for efficient interprocessor communications.

Traditional multiprocessor implementations have been very expensive in addition to being unable to exploit fully the available CPU horsepower. Multiprocessor systems have typically been built up from numerous board level or box level computers. The result is usually an immense amount of hardware with corresponding wiring, power consumption and communications problems. By the time the systems are interconnected, as much as 50 percent of the bus speed has been utilized just getting through the interfaces.

In addition, multiprocessor system software has been scarce. A multiprocessor system can easily be crippled by an inadequate load-sharing algorithm in the system software, which allows one CPU to do a great deal of work and the others to be idle. Great strides have been made recently in systems software, and even UNIX V.4 may be enhanced to support multiprocessing. Several commercial products from such manufacturers as DUAL Systems and UNISOFT do a credible job on 68030 type microprocessor systems now.

The microprocessor 310 architecture eliminates most of the interface friction, since up to 64 CPU 310/RAM 311 processors should be able to intercommunicate without buffers or latches. Each chip 312 has about 40 MIPS raw speed, because placing the DRAM 311 next to the CPU 310 allows the microprocessor 310 instruction cycle to be cut in half, compared to the microprocessor 50. A 64 chip array of these chips 312 is more powerful than any other existing computer. Such an array fits on a 3×5 card, cost less than a FAX machine, and draw about the same power as a small television.

Dramatic changes in price/performance always reshape existing applications and almost always create new ones. The introduction of microprocessors in the mid 1970s created video games, personal computers, automotive computers, electronically controlled appliances, and low cost computer peripherals.

The integrated circuit 312 will find applications in all of the above areas, plus create some new ones. A common generic parallel processing algorithm handles convolution/Fast Fourier Transform (FFT)/pattern recognition. Interesting product possibilities using the integrated circuit 312 include high speed reading machines, real-time speech recognition, spoken language translation, real-time robot vision, a product to identify people by their faces, and an automotive or aviation collision avoidance system.

A real time processor for enhancing high density television (HDTV) images, or compressing the HDTV information into a smaller bandwidth, would be very feasible. The load sharing in HDTV could be very straightforward. Splitting up the task according to color and frame would require 6, 9 or 12 processors. Practical implementation might require 4 meg RAMs integrated with the microprocessor 310.

The microprocessor 310 has the following specifications:

Control Lines
4—POWER/GROUND
1—CLOCK
32—DATA I/O
4—SYSTEM CONTROL
EXTERNAL MEMORY FETCH
EXTERNAL MEMORY FETCH AUTOINCREMENT X
EXTERNAL MEMORY FETCH AUTOINCREMENT Y
EXTERNAL MEMORY WRITE
EXTERNAL MEMORY WRITE AUTOINCREMENT X
EXTERNAL MEMORY WRITE AUTOINCREMENT Y
EXTERNAL PROM FETCH
LOAD ALL X REGISTERS
LOAD ALL Y REGISTERS
LOAD ALL PC REGISTERS
EXCHANGE X AND Y
INSTRUCTION FETCH
ADD TO PC
ADD TO X

WRITE MAPPING REGISTER
READ MAPPING REGISTER
REGISTER CONFIGURATION
MICROPROCESSOR 310 CPU 316 CORE
COLUMN LATCH1 (1024 BITS) 32×32 MUX
STACK POINTER (16 BITS)
COLUMN LATCH2 (1024 BITS) 32×32 MUX
RSTACK POINTER (16 BITS)
PROGRAM COUNTER 32 BITS
XO REGISTER 32 BITS (ACTIVATED ONLY FOR ON-CHIP ACCESSES)
YO REGISTER 32 BITS (ACTIVATED ONLY FOR ON-CHIP ACCESSES)
LOOP COUNTER 32 BITS
DMA CPU 314 CORE
DMA PROGRAM COUNTER 24 BITS
INSTRUCTION REGISTER 32 BITS
I/O & RAM ADDRESS REGISTER 32 BITS
Transfer Size Counter 12 Bits
Interval Counter 12 Bits

To offer memory expansion for the basic chip **312**, an intelligent DRAM can be produced This chip will be optimized for high speed operation with the integrated circuit **312** by having three on-chip address registers: Program Counter, X Register and Y register. As a result, to access the intelligent DRAM, no address is required, and a total access cycle could be as short as 10 nsec. Each expansion DRAM would maintain its own copy of the three registers and would be identified by a code specifying its memory address Incrementing and adding to the three registers will actually take place on the memory chips. A maximum of 64 intelligent DRAM peripherals would allow a large system to be created without sacrificing speed by introducing multiplexers or buffers.

There are certain differences between the microprocessor **310** and the microprocessor **50** that arise from providing the microprocessor **310** on the same die **312** with the DRAM **311**. Integrating the DRAM **311** allows architectural changes in the microprocessor **310** logic to take advantage of existing on-chip DRAM **311** circuitry. Row and column design is inherent in memory architecture. The DRAMs **311** access random bits in a memory array by first selecting a row of 1024 bits, storing them into a column latch, and then selecting one of the bits as the data to be read or written.

The time required to access the data is split between the row access and the column access Selecting data already stored in a column latch is faster than selecting a random bit by at least a factor of six. The microprocessor **310** takes advantage of this high speed by creating a number of column latches and using them as caches and shift registers. Selecting a new row of information may be thought of as performing a 1024-bit read or write with the resulting immense bus bandwidth.

1. The microprocessor **50** treats its 32-bit instruction register **108** (see FIGS 2 and 4) as a cache for four 8-bit instructions Since the DRAM **311** maintains a 1024-bit latch for the column bits, the microprocessor **310** treats the column latch as a cache for 128 8-bit instructions Therefore, the next instruction will almost always be already present in the cache. Long loops within the cache are also possible and more useful than the 4 instruction loops in the microprocessor **50**.

2. The microprocessor **50** uses two 16×32-bit deep register arrays **74** and **134** (FIG. 2) for the parameter stack and the return stack The microprocessor **310** creates two other 1024-bit column latches to provide the equivalent of two 32×32-bit arrays, which can be accessed twice as fast as a register array

3. The microprocessor **50** has a DMA capability which can be used for I/O to a video shift register. The microprocessor **310** uses yet another 1024-bit column latch as a long video shift register to drive a CRT display directly. For color displays, three on-chip shift registers could also be used. These shift registers can transfer pixels at a maximum of 100 MHz.

4. The microprocessor **50** accesses memory via an external 32-bit bus. Most of the memory **311** for the microprocessor **310** is on the same die **312**. External access to more memory is made using an 8-bit bus. The result is a smaller die, smaller package and lower power consumption than the microprocessor **50**.

5. The microprocessor **50** consumes about a third of its operating power charging and discharging the I/O pins and associated capacitances. The DRAMs **150** (FIG. 8) connected to the microprocessor **50** dissipate most of their power in the I/O drivers A microprocessor **310** system will consume about one-tenth the power of a microprocessor **50** system, since having the DRAM **311** next to the processor **310** eliminates most of the external capacitances to be charged and discharged.

6. Multiprocessing means splitting a computing task between numerous processors in order to speed up the solution The popularity of multiprocessing is limited by the expense of current individual processors as well as the limited interprocessor communications ability. The microprocessor **310** is an excellent multiprocessor candidate, since the chip **312** is a monolithic computer complete with memory, rendering it low-cost and physically compact

The shift registers implemented with the microprocessor **310** to perform video output can also be configured as interprocessor communication links. The INMOS transputer attempted a similar strategy, but at much lower speed and without the performance benefits inherent in the microprocessor **310** column latch architecture. Serial I/O is a prerequisite for many multiprocessor topologies because of the many neighbor processors which communicate. A cube has 6 neighbors. Each neighbor communicates using these lines:

DATA IN
CLOCK IN
READY FOR DATA
DATA OUT
DATA READY?
CLOCK OUT

A special start up sequence is used to initialize the on-chip DRAM **311** in each of the processors

The microprocessor **310** column latch architecture allows neighbor processors to deliver information directly to internal registers or even instruction caches of other chips **312**. This technique is not used with existing processors, because it only improves performance in a tightly coupled DRAM system.

7. The microprocessor **50** architecture offers two types of looping structures: LOOP-IF-DONE and MICRO-LOOP. The former takes an 8-bit to 24-bit operand to describe the entry point to the loop address. The latter performs a loop entirely within the 4 instruction queue and the loop entry point is implied as the first instruction in the queue. Loops entirely within the queue run without external instruction fetches and execute up to three times as fast as the long loop construct The microprocessor **310** retains both constructs with a few differences. The microprocessor **310** microloop

functions in the same fashion as the microprocessor **50** operation, except the queue is 1024-bits or 128 8-bit instructions long. The microprocessor **310** microloop can therefore contain jumps, branches, calls and immediate operations not possible in the 4 8-bit instruction microprocessor **50** queue.

Microloops in the microprocessor **50** can only perform simple block move and compare functions. The larger microprocessor **310** queue allows entire digital signal processing or floating point algorithms to loop at high speed in the queue.

The microprocessor **50** offers four instructions to redirect execution:

CALL
BRANCH
BRANCH-IF-ZERO
LOOP-IF-NOT-DONE

These instructions take a variable length address operand 8, 16 or 24 bits long. The microprocessor **50** next address logic treats the three operands similarly by adding or subtracting them to the current program counter. For the microprocessor **310**, the 16 and 24-bit operands function in the same manner as the 16 and 24-bit operands in the microprocessor **50**. The 8-bit class operands are reserved to operate entirely within the instruction queue. Next address decisions can therefore be made quickly, because only 10 bits of addresses are affected, rather than **32**. There is no carry or borrow generated past the 10 bits.

8. The microprocessor **310** CPU **316** resides on an already crowded DRAM die **312**. To keep chip size as small as possible, the DMA processor **72** of the microprocessor **50** has been replaced with a more traditional DMA controller **314**. DMA is used with the microprocessor **310** to perform the following functions:

Video output to a CRT
Multiprocessor serial communications
8-bit parallel I/O

The DMA controller **314** can maintain both serial and parallel transfers simultaneously. The following DMA sources and destinations are supported by the microprocessor **310**:

DESCRIPTION	I/O	LINES
1. Video shift register	OUTPUT	1 to 3
2. Multiprocessor serial	BOTH	6 lines/channel
3. 8-bit parallel	BOTH	8 data, 4 control

The three sources use separate 1024-bit buffers and separate I/O pins. Therefore, all three may be active simultaneously without interference.

The microprocessor **310** can be implemented with either a single multiprocessor serial buffer or separate receive and sending buffers for each channel, allowing simultaneous bidirectional communications with six neighbors simultaneously.

FIGS. **10** and **11** provide details of the PROM DMA used in the microprocessor **50**. The microprocessor **50** executes faster than all but the fastest PROMs. PROMS are used in a microprocessor **50** system to store program segments and perhaps entire programs. The microprocessor **50** provides a feature on power-up to allow programs to be loaded from low-cost, slow speed PROMs into high speed DRAM for execution. The logic which performs this function is part of the DMA memory controller **118**. The operation is similar to DMA, but not identical, since four 8-bit bytes must be assembled on the microprocessor **50** chip, then written to the DRAM **150**.

The microprocessor **50** directly interfaces to DRAM **150** over a triple multiplexed data and address bus **350**, which carries RAS addresses, CAS addresses and data. The EPROM **260**, on the other hand, is read with non-multiplexed busses. The microprocessor **50** therefore has a special mode which unmultiplexes the data and address lines to read 8 bits of EPROM data. Four 8-bit bytes are read in this fashion. The multiplexed bus **350** is turned back on, and the data is written to the DRAM **150**.

When the microprocessor **50** detects a RESET condition, the processor stops the main CPU **70** and forces a mode **0** (PROM LOAD) instruction into the DMA CPU **72** instruction register. The DMA instruction directs the memory controller to read the EPROM **260** data at 8 times the normal access time for memory. Assuming a 50 MHz microprocessor **50**, this means an access time of 320 nsec. The instruction also indicates:

The selection address of the EPROM **260** to be loaded.
The number of 32-bit words to transfer,
The DRAM **150** address to transfer into.

The sequence of activities to transfer one 32-bit word from EPROM **260** to DRAM **150** are:

1. RAS goes low at **352**, latching the EPROM **260** select information from the high order address bits. The EPROM **260** is selected.
2. Twelve address bits (consisting of what is normally DRAM CAS addresses plus two byte select bits are placed on the bus **350** going to the EPROM **260** address pins. These signals will remain on the lines until the data from the EPROM **260** has been read into the microprocessor **50**. For the first byte, the byte select bits will be binary **00**.
3. CAS goes low at **354**, enabling the EPROM **260** data onto the lower 8 bits of the external address/data bus **350**. NOTE: It is important to recognize that, during this part of the cycle, the lower 8 bits of the external data/address bus are functioning as inputs, but the rest of the bus is still acting as outputs.
4. The microprocessor **50** latches these eight least significant bits internally and shifts them 8 bits left to shift them to the next significant byte position.
5. Steps **2, 3** and **4** are repeated with byte address **01**.
6. Steps **2, 3** and **4** are repeated with byte address **10**.
7. Steps **2, 3** and **4** are repeated with byte address **11**.
8. CAS goes high at **356**, taking the EPROM **260** off the data bus.
9. RAS goes high at **358**, indicating the end of the EPROM **260** access.
10. RAS goes low at **360**, latching the DRAM select information from the high order address bits. At the same time, the RAS address bits are latched into the DRAM **150**. The DRAM **150** is selected.
11. CAS goes low at **362**, latching the DRAM **150** CAS addresses.
12. The microprocessor **50** places the previously latched EPROM **260** 32-bit data onto the external address/data bus **350**. W goes low at **364**, writing the 32 bits into the DRAM **150**.
13. W goes high at **366**. CAS goes high at **368**. The process continues with the next word.

FIG. **12** shows details of the microprocessor **50** memory controller **118**. In operation, bus requests stay present until they are serviced. CPU **70** requests are priorit-

ized at 370 in the order of: 1, Parameter Stack; 2, Return Stack; 3, Data Fetch; 4, Instruction Fetch The resulting CPU request signal and a DMA request signal are supplied as bus requests to bus control 372, which provides a bus grant signal at 374. Internal address bus 136 and a DMA counter 376 provide inputs to a multiplexer 378. Either a row address or a column address are provided as an output to multiplexed address bus 380 as an output from the multiplexer 378 The multiplexed address bus 380 and the internal data bus 90 provide address and data inputs, respectively, to multiplexer 382. Shift register 384 supplies row address strobe (RAS) 1 and 2 control signals to multiplexer 386 and column address strobe (CAS) 1 and 2 control signals to multiplexer 388 on lines 390 and 392. The shift register 384 also supplies output enable (OE) and write (W) signals on lines 394 and 396 and a control signal on line 398 to multiplexer 382. The shift register 384 receives a RUN signal on line 400 to generate a memory cycle and supplies a MEMORY READY signal on line 402 when an access is complete

Stack/Register Architecture

Most microprocessors use on-chip registers for temporary storage of variables The on-chip registers access data faster than off-chip RAM A few microprocessors use an on-chip push down stack for temporary storage.

A stack has the advantage of faster operation compared to on-chip registers by avoiding the necessity to select source and destination registers (A math or logic operation always uses the top two stack items as source and the top of stack as destination.) The stack's disadvantage is that it makes some operations clumsy. Some compiler activities in particular require on-chip registers for efficiency.

As shown in FIG 13, the microprocessor 50 provides both on-chip registers 134 and a stack 74 and reaps the benefits of both.

Benefits

1 Stack math and logic is twice as fast as those available on an equivalent register only machine Most programmers and optimizing compilers can take advantage of this feature.

2 Sixteen registers are available for on-chip storage of local variables which can transfer to the stack for computation The accessing of variables is three to four times as fast as available on a strictly stack machine.

The combined stack 74/register 134 architecture has not been used previously due to inadequate understanding by computer designers of optimizing compilers and the mix of transfer versus math/logic instructions.

ADAPTIVE MEMORY CONTROLLER

A microprocessor must be designed to work with small or large memory configurations. As more memory loads are added to the data, address, and control lines, the switching speed of the signals slows down The microprocessor 50 multiplexes the address/data bus three ways, so timing between the phases is critical. A traditional approach to the problem allocates a wide margin of time between bus phases so that systems will work with small or large numbers of memory chips connected A speed compromise of as much as 50% is required

As shown in FIG 14, the microprocessor 50 uses a feedback technique to allow the processor to adjust memory bus timing to be fast with small loads and slower with large ones. The OUTPUT ENABLE (OE) line 152 from the microprocessor 50 is connected to all memories 150 on the circuit board The loading on the output enable line 152 to the microprocessor 50 is directly related to the number of memories 150 connected. By monitoring how rapidly OE 152 goes high after a read, the microprocessor 50 is able to determine when the data hold time has been satisfied and place the next address on the bus.

The level of the OE line 152 is monitored by CMOS input buffer 410 which generates an internal READY signal on line 412 to the microprocessor's memory controller. Curves 414 and 416 of the FIG 15 graph show the difference in rise time likely to be encountered from a lightly to heavily loaded memory system When the OE line 152 has reached a predetermined level to generate the READY signal, driver 418 generates an OUTPUT ENABLE signal on OE line 152.

Skip Within The Instruction Cache

The microprocessor 50 fetches four 8-bit instructions each memory cycle and stores them in a 32-bit instruction register 108, as shown in FIG. 16. A class of "test and skip" instructions can very rapidly execute a very fast jump operation within the four instruction cache

Skip Conditions

Always
ACC non-zero
ACC negative
Carry flag equal logic one
Never
ACC equal zero
ACC positive
Carry flag equal logic zero

The SKIP instruction can be located in any of the four byte positions 420 in the 32-bit instruction register 108 If the test is successful, SKIP will jump over the remaining one, two, or three 8-bit instructions in the instruct ion register 108 and cause the next four-instruction group to be loaded into the register 108. As shown, the SKIP operation is implemented by resetting the 2-bit microinstruction counter 180 to zero on line 422 and simultaneously latching the next instruction group into the register 108. Any instructions following the SKIP in the instruction register are overwritten by the new instructions and not executed

The advantage of SKIP is that optimizing compilers and smart programmers can often use it in place of the longer conditional JUMP instruction. SKIP also makes possible microloops which exit when the loop counts down or when the SKIP jumps to the next instruction group The result is very fast code.

Other machines (such as the PDP-8 and Data General NOVA) provide the ability to skip a single instruction. The microprocessor 50 provides the ability to skip up to three instructions

Microloop In The Instruction Cache

The microprocessor 50 provides the MICROLOOP instruction to execute repetitively from one to three instructions residing in the instruction register 108. The microloop instruction works in conjunction with the LOOP COUNTER 92 (FIG 2) connected to the internal data bus 90. To execute a microloop, the program stores a count in LOOP COUNTER 92. MICROLOOP may be placed in the first, second, third, or last byte 420 of the instruction register 108 If placed in the first position, execution will just create a delay equal to the

number stored in LOOP COUNTER **92** times the machine cycle. If placed in the second, third, or last byte **420**, when the microloop instruction is executed, it will test the LOOP COUNT for zero. If zero, execution will continue with the next instruction. If not zero, the LOOP COUNTER **92** is decremented and the 2-bit microinstruction counter is cleared, causing the preceding instructions in the instruction register to be executed again.

Microloop is useful for block move and search operations. By executing a block move completely out of the instruction register **108**, the speed of the move is doubled, since all memory cycles are used by the move rather than being shared with instruction fetching. Such a hardware implementation of microloops is much faster than conventional software implementation of a comparable function.

Optimal CPU Clock Scheme

The designer of a high speed microprocessor must produce a product which operate over wide temperature ranges, wide voltage swings, and wide variations in semiconductor processing. Temperature, voltage, and process all affect transistor propagation delays. Traditional CPU designs are done so that with the worse case of the three parameters, the circuit will function at the rated clock speed. The result are designs that must be clocked a factor of two slower than their maximum theoretical performance, so they will operate properly in worse case conditions.

The microprocessor **50** uses the technique shown in FIGS. **17-19** to generate the system clock and its required phases. Clock circuit **430** is the familiar "ring oscillator" used to test process performance. The clock is fabricated on the same silicon chip as the rest of the microprocessor **50**

The ring oscillator frequency is determined by the parameters of temperature, voltage, and process. At room temperature, the frequency will be in the neighborhood of 100 MHZ. At 70 degrees Centigrade, the speed will be 50 MHZ. The ring oscillator **430** is useful as a system clock, with its stages **431** producing phase O-phase 3 outputs **433** shown in FIG. **19**, because its performance tracks the parameters which similarly affect all other transistors on the same silicon die. By deriving system timing from the ring oscillator **430**, CPU **70** will always execute at the maximum frequency possible, but never too fast. For example, if the processing of a particular die is not good resulting in slow transistors, the latches and gates on the microprocessor **50** will operate slower than normal. Since the microprocessor **50** ring oscillator clock **430** is made from the same transistors on the same die as the latches and gates, it too will operate slower (oscillating at a lower frequency), providing compensation which allows the rest of the chip's logic to operate properly.

Asynchronous/Synchronous CPU

Most microprocessors derive all system timing from a single clock. The disadvantage is that different parts of the system can slow all operations. The microprocessor **50** provides a dual-clock scheme as shown in FIG. **17**, with the CPU **70** operating asynchronously to I/O interface **432** forming part of memory controller **118** (FIG. 2) and the I/O interface **432** operating synchronously with the external world of memory and I/O devices. The CPU **70** executes at the fastest speed possible using the adaptive ring counter clock **430**. Speed may vary by a factor of four depending upon temperature, voltage, and process. The external world must be synchronized to the microprocessor **50** for operations such as video display updating and disc drive reading and writing. This synchronization is performed by the I/O interface **432**, speed of which is controlled by a conventional crystal clock **434**. The interface **432** processes requests for memory accesses from the microprocessor **50** and acknowledges the presence of I/O data. The microprocessor **50** fetches up to four instructions in a single memory cycle and can perform much useful work before requiring another memory access. By decoupling the variable speed of the CPU **70** from the fixed speed of the I/O interface **432**, optimum performance can be achieved by each. Recoupling between the CPU **70** and the interface **432** is accomplished with handshake signals on lines **436**, with data/addresses passing on bus **90**, **136**.

Asynchronous/Synchronous CPU Imbedded On A Dram Chip

System performance is enhanced even more when the DRAM **311** and CPU **314** (FIG. 9) are located on the same die. The proximity of the transistors means that DRAM **311** and CPU **314** parameters will closely follow each other. At room temperature, not only would the CPU **314** execute at 100 MHZ but the DRAM **311** would access fast enough to keep up. The synchronization performed by the I/O interface **432** would be for DMA and reading and writing I/O ports. In some systems (such as calculators) no I/O synchronization at all would be required, and the I/O clock would be tied to the ring counter clock.

Variable Width Operands

Many microprocessors provide variable width operands. The microprocessor **50** handles operands of **8**, **16**, or **24** bits using the same op-code. FIG. **20** shows the 32-bit instruction register **108** and the 2-bit microinstruction register **180** which selects the 8-bit instruction. Two classes of microprocessor **50** instructions can be greater than 8-bits, JUMP class and IMMEDIATE. A JUMP or IMMEDIATE op-code is 8-bits, but the operand can be 8, 16, or 24 bits long. This magic is possible because operands must be right justified in the instruction register. This means that the least significant bit of the operand is always located in the least significant bit of the instruction register. The microinstruction counter **180** selects which 8-bit instruction to execute. If a JUMP or IMMEDIATE instruction is decoded, the state of the 2-bit microinstruction counter selects the required 8, 16, or 24 bit operand onto the address or data bus. The unselected 8-bit bytes are loaded with zeros by operation of decoder **440** and gates **442**. The advantage of this technique is the saving of a number of op-codes required to specify the different operand sizes in other microprocessors.

Triple Stack Cache

Computer performance is directly related to the system memory bandwidth. The faster the memories, the faster the computer. Fast memories are expensive, so techniques have been developed to move a small amount of high-speed memory around to the memory addresses where it is needed. A large amount of slow memory is constantly updated by the fast memory, giving the appearance of a large fast memory array. A common implementation of the technique is known as a high-speed memory cache. The cache may be thought

of as fast acting shock absorber smoothing out the bumps in memory access. When more memory is required than the shock can absorb, it bottoms out and slow speed memory is accessed. Most memory operations can be handled by the shock absorber itself.

The microprocessor **50** architecture has the ALU **80** (FIG. 2) directly coupled to the top two stack locations **76** and **78**. The access time of the stack **74** therefore directly affects the execution speed of the processor. The microprocessor **50** stack architecture is particularly suitable to a triple cache technique, shown in FIG. **21** which offers the appearance of a large stack memory operating at the speed of on-chip latches **450**. Latches **450** are the fastest form of memory device built on the chip, delivering data in as little as 3 nsec. However latches **450** require large numbers of transistors to construct. On-chip RAM **452** requires fewer transistors than latches, but is slower by a factor of five (15 nsec access). Off-chip RAM **150** is the slowest storage of all. The microprocessor **50** organizes the stack memory hierarchy as three interconnected stacks **450**, **452** and **454**. The latch stack **450** is the fastest and most frequently used. The on-chip RAM stack **452** is next. The off-chip RAM stack **454** is slowest. The stack modulation determines the effective access time of the stack. If a group of stack operations never push or pull more than four consecutive items on the stack, operations will be entirely performed in the 3 nsec latch stack. When the four latches **456** are filled. the data in the bottom of the latch stack **450** is written to the top of the on-chip RAM stack **452**. When the sixteen locations **458** in the on-chip RAM stack **452** are filled, the data in the bottom of the on-chip RAM stack **452** is written to the top of the off-chip RAM stack **454**. When popping data off a full stack **450**, four pops will be performed before stack empty line **460** from the latch stack pointer **462** transfers data from the on-chip RAM stack **452**. By waiting for the latch stack **450** to empty before performing the slower on-chip RAM access, the high effective speed of the latches **456** are made available to the processor. The same approach is employed with the on-chip RAM stack **452** and the off-chip RAM stack **454**.

Polynomial Generation Instruction

Polynomials are useful for error correction, encryption, data compression, and fractal generation. A polynomial is generated by a sequence of shift and exclusive OR operations. Special chips are provided for this purpose in the prior art.

The microprocessor **50** is able to generate polynomials at high speed without external hardware by slightly modifying how the ALU **80** works. As shown in FIG. **22**, a polynomial is generated by loading the "order" (also known as the feedback terms) into C Register **470**. The value thirty one (resulting in 32 is loaded into DOWN COUNTER **472**. A register **474** is loaded with zero. B register **476** is loaded with the starting polynomial value. When the POLY instruction executes, C register **470** is exclusively ORed with A register **474** if the least significant bit of B register **476** is a one. Otherwise, the contents of the A register **474** passes through the ALU **80** unaltered. The combination of A and B is then shifted right (divided by **2**) with shifters **478** and **480**. The operation automatically repeats the specified number of iterations, and the resulting polynomial is left in A register **474**.

Fast Multiply

Most microprocessors offer a 16×16 or 32×32 bit multiply instruction. Multiply when performed sequentially takes one shift/add per bit, or 32 cycles for 32 bit data. The microprocessor **50** provides a high speed multiply which allows multiplication by small numbers using only a small number of cycles. FIG. **23** shows the logic used to implement the high speed algorithm. To perform a multiply, the size of the multiplier less one is placed in the DOWN COUNTER **472**. For a four bit multiplier, the number three would be stored in the DOWN COUNTER **472**. Zero is loaded into the A register **474**. The multiplier is written bit reversed into the B Register **476**. For example, a bit reversed five (binary **0101**) would be written into B as **1010**. The multiplicand is written into the C register **470**. Executing the FAST MULT instruction will leave the result in the A Register **474**, when the count has been completed. The fast multiply instruction is important because many applications scale one number by a much smaller number. The difference in speed between multiplying a 32×32 bit and a 32×4 bit is a factor of **8**. If the least significant bit of the multiplier is a "ONE" the contents of the A register **474** and the C register **470** are added. If the least significant bit of the multiplier is a "ZERO" the contents of the A register are passed through the ALU **80** unaltered. The output of the ALU **80** is shifted left by shifter **482** in each iteration. The contents of the B register **476** are shifted right by the shifter **480** in each iteration.

INSTRUCTION EXECUTION PHILOSOPHY

The microprocessor **50** uses high speed D latches in most of the speed critical areas. Slower on-chip RAM is used as secondary storage.

The microprocessor **50** philosophy of instruction **30** execution is to create a hierarchy of speed as follows:

Logic and D latch transfers	1 cycle	20 nsec
Math	2 cycles	40 nsec
Fetch/store on-chip RAM	2 cycles	40 nsec
Fetch store in current RAS page	4 cycles	80 nsec
Fetch/store with RAS cycle	11 cycles	220 nsec

With a 50 MHZ clock, many operations can be performed in 20 nsec. and almost everything else in 40 nsec.

To maximize speed, certain techniques in processor design have been used. They include:

- Eliminating arithmetic operations on addresses,
- Fetching up to four instructions per memory cycle,
- Pipelineless instruction decoding
- Generating results before they are needed,
- Use of three level stack caching

PIPELINE PHILOSOPHY

Computer instructions are usually broken down into sequential pieces, for example: fetch, decode, register read, execute, and store. Each piece will require a single machine cycle. In most Reduced Instruction Set Computer (RISC) chips, instruction require from three to six cycles.

RISC instructions are very parallel. For example, each of **70** different instructions in the SPARC (SUN Computer's RISC chip) has five cycles. Using a technique called "pipelining", the different phases of consecutive instructions can be overlapped.

To understand pipelining, think of building five residential homes. Each home will require in sequence, a foundation, framing, plumbing and wiring, roofing, and interior finish. Assume that each activity takes one week. To build one house will take five weeks.

But what if you want to build an entire subdivision? You have only one of each work crew, but when the foundation men finish on the first house, you immediately start them on the second one, and so on. At the end of five weeks, the first home is complete, but you also have five foundations. If you have kept the framing, plumbing, roofing, and interior guys all busy, from five weeks on, a new house will be completed each week.

This is the way a RISC chip like SPARC appears to execute an instruction in a single machine cycle. In reality, a RISC chip is executing one fifth of five instructions each machine cycle. And if five instructions stay in sequence, an instruction will be completed each machine cycle.

The problems with a pipeline are keeping the pipe full with instructions. Each time an out of sequence instruction such as a BRANCH or CALL occurs, the pipe must be refilled with the next sequence. The resulting dead time to refill the pipeline can become substantial when many IF/THEN/ELSE statements or subroutines are encountered.

The Pipeline Approach

The microprocessor **50** has no pipeline as such. The approach of this microprocessor to speed is to overlap instruction fetching with execution of the previously fetched instruction(s). Beyond that, over half the instructions (the most common ones) execute entirely in a single machine cycle of 20 nsec. This is possible because:

1. Instruction decoding resolves in 2.5 nsec.
2. Incremented/decremented and some math values are calculated before they are needed, requiring only a latching signal to execute.
3. Slower memory is hidden from high speed operations by high-speed D latches which access in 4 nsec.

The disadvantage for this microprocessor is a more complex chip design process. The advantage for the chip user is faster ultimate throughput since pipeline stalls cannot exist. Pipeline synchronization with availability flag bits and other such pipeline handling is not required by this microprocessor.

For example, in some RISC machines an instruction which tests a status flag may have to wait for up to four cycles for the flag set by the previous instruction to be available to be tested. Hardware and software debugging is also somewhat easier because the user doesn't have to visualize five instructions simultaneously in the pipe.

OVERLAPPING INSTRUCTION FETCH/EXECUTE

The slowest procedure the microprocessor **50** performs is to access memory. Memory is accessed when data is read or written. Memory is also read when instructions are fetched. The microprocessor **50** is able to hide fetch of the next instruction behind the execution of the previously fetched instruction(s). The microprocessor **50** fetches instructions in 4-byte instruction groups. An instruction group may contain from one to four instructions. The amount of time required to execute the instruction group ranges from 4 cycles for simple instructions to 64 cycles for a multiply.

When a new instruction group is fetched, the microprocessor instruction decoder looks at the most significant bit of all four of the bytes. The most significant bit of an instruction determines if a memory access is required. For example, CALL, FETCH, and STORE all require a memory access to execute. If all four bytes have nonzero most significant bits, the microprocessor initiates the memory fetch of the next sequential 4-byte instruction group. When the last instruction in the group finishes executing, the next 4-byte instruction group is ready and waiting on the data bus needing only to be latched into the instruction register. If the 4-byte instruction group required four or more cycles to execute and the next sequential access was a column address strobe (CAS) cycle, the instruction fetch was completely overlapped with execution.

INTERNAL ARCHITECTURE

The microprocessor **50** architecture consists of the following:

PARAMETER STACK	<--> ALU* <-->	Y REGISTER RETURN STACK
<---32 BITS---> 16 DEEP Used for math and logic		<---32 BITS---> 16 DEEP Used for subroutine and interrupt return addresses as well as local variables.
Push down stack Can overflow into off-chip RAM		Push down stack Can overflow into off-chip RAM Can also be accessed relative to top of stack.
LOOP COUNTER	(32-bits, can decrement by 1) Used by class of test and loop instructions.	
X REGISTER	(32-bits, can increment or decrement by 4). Used to point to RAM locations.	
PROGRAM COUNTER	(32-bits, increments by 4) Points to 4-byte instruction groups in RAM	
INSTRUCTION REG	(32-Bits). Holds 4-byte instruction groups while they are being decoded	

-continued

and executed.

*Math and logic operations use the TOP item and NEXT to top Parameter Stack items as the operands The result is pushed onto the Parameter Stack.
*Return addresses from subroutines are placed on the Return Stack. The Y REGISTER is used as a pointer to RAM locations Since the Y REGISTER is the top item of the Return Stack nesting of indices is straightforward

MODE—A register with mode and status bits.

Mode-Bits

Slow down memory accesses by 8 if "1". Run full speed if "0" (Provided for access to slow EPROM.)

Divide the system clock by 1023 if "1" to reduce power consumption. Run full speed if "0" (On-chip counters slow down if this bit is set.)

Enable external interrupt 1.
Enable external interrupt 2.
Enable external interrupt 3.
Enable external interrupt 4.
Enable external interrupt 5.
Enable external interrupt 6.
Enable external interrupt 7.

On-Chip Memory Locations

MODE-BITS
DMA-POINTER
DMA-COUNTER
STACK-POINTER—Pointer into Parameter Stack
STACK-DEPTH—Depth of on-chip Parameter Stack
RSTACK-POINTER—Pointer into Return Stack
RSTACK-DEPTH—Depth of on-chip Return Stack

Addressing Mode High Points

The data bus is 32-bits wide. All memory fetches and stores are 32-bits. Memory bus addresses are 30 bits. The least significant 2 bits are used to select one-of-four bytes in some addressing modes. The Program Counter, X Register, and Y Register are implemented as D latches with their outputs going to the memory address bus and the bus incrementer/decrementer. Incrementing one of these registers can happen quickly, because the incremented value has already rippled through the inc/dec logic and need only be clocked into the latch. Branches and Calls are made to 32-bit word boundaries.

INSTRUCTION SET

32-Bit Instruction Format

The thirty two bit instructions are CALL, BRANCH, BRANCH-IF-ZERO, and LOOP-IF-NOT-DONE. These instructions require the calculation of an effective address. In many computers, the effective address is calculated by adding or subtracting an operand with the current Program Counter. This math operation requires from four to seven machine cycles to perform and can definitely bog down machine execution. The microprocessor's strategy is to perform the required math operation at assembly or linking time and do a much simpler "Increment to next page" or "Decrement to previous page" operation at run time. As a result, the microprocessor branches execute in a single cycle.

24-Bit Operand Form

Byte 1	Byte 2	Byte 3	Byte 4
WWWWWW XX -	YYYYYYYY -	YYYYYYYY -	YYYYYYYY

With a 24-bit operand, the current page is considered to be defined by the most significant 6 bits of the Program Counter.

16-Bit Operand Form

QQQQQQQQ—Wwwwww

XX—YYYYYYYY—YYYYYYYY With a 16-bit operand, the current page is considered to be defined by the most significant 14 bits of the Program Counter.

8-Bit Operand Form

QQQQQQQQ—QQQQQQQQ—WwWwWw

XX—YYYYYYYY With an 8-bit operand, the current page is considered to be defined by the most significant 22 bits of the Program Counter.

QQQQQQQQ—Any 8-bit instruction

WWWWWW—Instruction op-code.

XX—Select how the address bits will be used:

00 - Make all high-order bits zero. (Page zero addressing)

01 - Increment the high-order bits. (Use next page)

10 - Decrement the high-order bits. (Use previous page)

11 - Leave the high-order bits unchanged. (Use current page)

YYYYYYYY - The address operand field. This field is always shifted left two bits (to generate a word rather than byte address) and loaded into the Program Counter. The microprocessor instruction decoder figures out the width of the operand field by the location of the instruction op-code in the four bytes.

The compiler or assembler will normally use the shortest operand required to reach the desired address so that the leading bytes can be used to hold other instructions. The effective address is calculated by combining:

The current Program Counter,

The 8, 16, or 24 bit address operand in the instruction,

Using one of the four allowed addressing modes.

EXAMPLES OF EFFECTIVE ADDRESS CALCULATION

Example 1

Byte 1	Byte 2	Byte 3	Byte 4
QQQQQQQQ	QQQQQQQQ	00000011	10011000

The "QQQQQQQQs" in Byte 1 and 2 indicate space in the 4-byte memory fetch which could be hold two other instructions to be executed prior to the CALL instruction. Byte 3 indicates a CALL instruction (six zeros) in the current page (indicated by the 11 bits). Byte 4 indicates that the hexadecimal number 98 will be forced into the Program Counter bits 2 through 10 (Remember, a CALL or BRANCH always goes to a

word boundary so the two least significant bits are always set to zero). The effect of this instruction would be to CALL a subroutine at WORD location HEX 98 in the current page. The most significant 22 bits of the Program Counter define the current page and will be unchanged

Example 2

Byte 1	Byte 2	Byte 3	Byte 4
000001 01	00000001	00000000	00000000

If we assume that the Program Counter was HEX **0000 0156** which is binary:

00000000 00000000 00000001 01010110=OLD PROGRAM COUNTER.

Byte 1 indicates a BRANCH instruction op code (**000001**) and "**01**" indicates select the next page. Byte 2,3, and 4 are the address operand. These 24-bits will be shifted to the left two places to define a WORD address. HEX **0156** shifted left two places is HEX **0558**. Since this is a 24-bit operand instruction, the most significant 6 bits of the Program Counter define the current page. These six bits will be incremented to select the next page. Executing this instruction will cause the Program Counter to be loaded with HEX **0400 0558** which is binary:

00000100 00000000 00000101 01011000=NEW PROGRAM COUNTER.

Instructions

Call-Long

0000 00XX—YYYYYYYY—YYYYYYYY—YYYYYYYY

Load the Program Counter with the effective WORD address specified. Push the current PC contents onto the RETURN STACK.

Other Effects: CARRY or modes, no effect. May cause Return Stack to force an external memory cycle if on-chip Return Stack is full.

Branch

0000 01XX—YYYYYYYY—Yyyyyyyy—yyyyyyyy

Load the Program Counter with the effective WORD address specified.

Other Effects: NONE

Branch-If-Zone

0000 10XX—YYYYYYYY—yyyyyyyy—yyyyyyyy

Test the TOP value on the Parameter Stack. If the value is equal to zero, load the Program Counter with the effective WORD address specified. If the TOP value is not equal to zero, increment the Program Counter and fetch and execute the next instruction.

Other Effects: NONE

Loop-If-Not-Done

0000 11YY—(XXXX XXXX)—(XXXX XXXX)—(XXXX XXXX)

If the LOOP COUNTER is not zero, load the Program Counter with the effective WORD address specified. If the LOOP COUNTER is zero, decrement the LOOP COUNTER, increment the Program Counter and fetch and execute the next instruction.

Other Effects: NONE

8-Bit Instructions Philosophy

Most of the work in the microprocessor **50** is done by the 8-bit instructions. Eight bit instructions are possible with the microprocessor because of the extensive use of implied stack addressing. Many 32-bit architectures use

8-bits to specify the operation to perform but use an additional 24-bits to specify two sources and a destination.

For math and logic operations, the microprocessor **50** exploits the inherent advantage of a stack by designating the source operand(s) as the top stack item and the next stack item. The math or logic operation is performed, the operands are popped from the stack, and the result is pushed back on the stack. The result is a very efficient utilization of instruction bits as well as registers. A comparable situation exists between Hewlett Packard calculators (which use a stack) and Texas Instrument calculators which don't. The identical operation on an HP will require one half to one third the keystrokes of the TI.

The availability of 8-bit instructions also allows another architectural innovation, the fetching of four instructions in a single 32-bit memory cycle. The advantages of fetching multiple instructions are:

Increased execution speed even with slow memories,

Similar performance to the Harvard (separate data and

instruction busses) without the expense,

Opportunities to optimize groups of instructions,

The capability to perform loops within this mini-cache.

The microloops inside the four instruction group are effective for searches and block moves

Skip Instructions

The microprocessor **50** fetches instructions in 32-bit chunks called 4-byte instruction groups. These four bytes may contain four 8-bit instructions or some mix of 8-bit and 16 or 24-bit instructions. SKIP instructions in the microprocessor skip any remaining instructions in a 4-byte instruction group and cause a memory fetch to get the next 4-byte instruction group. Conditional SKIPs when combined with 3-byte BRANCHES will create conditional BRANCHES. SKIPs may also be used in situations when no use can be made of the remaining bytes in a 4-instruction group. A SKIP executes in a single cycle, whereas a group of three NOPs would take three cycles.

Skip-Always—skip any remaining instructions in this 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group.

Skip-If-Zero—If the TOP item of the Parameter Stack is zero, skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the TOP item is not zero, execute the next sequential instruction.

Skip-If-Positive—If the TOP item of the Parameter Stack has a the most significant bit (the sign bit) equal to "**0**", skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the TOP item is not "**0**" execute the next sequential instruction

Skip-If-No-Carry—If the CARRY flag from a SHIFT or arithmetic operation is not equal to "**1**", skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to

fetch the next 4-byte instruction group. If the CARRY is equal to "1", execute the next sequential instruction.

Skip-Never Execute the next sequential (NOP) instruction (Delay one machine cycle)

Skip-If-Not-Zero—If the TOP item on the Parameter Stack is not equal to "0" skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the TOP item is equal 0" execute the next sequential instruction

Skip-If-Negative—If the TOP item on the Parameter Stack has its most significant bit (sign bit) set to "1", skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the TOP item has its most significant bit set to "0" execute the next sequential instruction.

Skip-If-Carry—If the CARRY flag is set to "1" as a result of SHIFT or arithmetic operation, skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the CARRY flag is "0" execute the next sequential instruction

Microloops

Microloops are a unique feature of the microprocessor architecture which allows controlled looping within a 4-byte instruction group. A microloop instruction tests the LOOP COUNTER for "0" and may perform an additional test. If the LOOP COUNTER is not "0" and the test is met, instruction execution continues with the first instruction in the 4-byte instruction group, and the LOOP COUNTER is decremented. A microloop instruction will usually be the last byte in a 4-byte instruction group, but it can be any byte. If the LOOP COUNTER is "0" or the test is not met, instruction execution continues with the next instruction. If the microloop is the last byte in the 4-byte instruction group, the most significant 30-bits of the Program Counter are incremented and the next 4-byte instruction group is fetched from memory. On a termination of the loop on LOOP COUNTER equal to "0" the LOOP COUNTER will remain at "0". Microloops allow short iterative work such as moves and searches to be performed without slowing down to fetch instructions from memory

Example

Byte 1	Byte 2
FETCH-VIA-X-AUTOINCREMENT	STORE-VIA-Y-AUTOINCREMENT
Byte 3	Byte 4
ULOOP-UNTIL-DONE	QQQQQQQQ

This example will perform a block move. To initiate the transfer, X will be loaded with the starting address of the source. Y will be loaded with the starting address of the destination. The LOOP COUNTER will be loaded with the number of 32-bit words to move. The microloop will FETCH and STORE and count down the LOOP COUNTER until it reaches zero QQQQQQQQ indicates any instruction can follow

Microloop Instructions

ULOOP-UNTIL-DONE—If the LOOP COUNTER is not "0", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" continue execution with the next instruction.

ULOOP-IF-ZERO—If the LOOP COUNTER is not "0" and the TOP item on the Parameter Stack is "0", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the TOP item is "1" continue execution with the next instruction

Uloop-if-positive—If the LOOP COUNTER is not "0" and the most significant bit (sign bit) is "0" continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the TOP item is "1", continue execution with the next instruction

Uloop-if-not-carry-clear—If the LOOP COUNTER is not "0" and the floating point exponents found in TOP and NEXT are not aligned, continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the exponents are aligned, continue execution with the next instruction. This instruction is specifically designed for combination with special SHIFT instructions to align two floating point numbers

ULoop-Never—(DECREMENT-LOOP-COUNTER) Decrement the LOOP COUNTER. Continue execution with the next instruction.

ULoop-If-Not-Zero—If the LOOP COUNTER is not "0" and the TOP item of the Parameter Stack is "0", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the TOP item is "1", continue execution with the next instruction.

ULoop-If-Negative—If the LOOP COUNTER is not "0" and the most significant bit (sign bit) of the TOP item of the Parameter Stack is "1", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the most significant bit of the Parameter Stack is "0" continue execution with the next instruction

ULoop-If-Carry-Set—If the LOOP COUNTER is not "0" and the exponents of the floating point numbers found in TOP and NEXT are not aligned, continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the exponents are aligned, continue execution with the next instruction.

Return From Subroutine Or Interrupt

Subroutine calls and interrupt acknowledgements cause a redirection of normal program execution. In both cases, the current Program Counter is pushed onto the Return Stack, so the microprocessor can return to its place in the program after executing the subroutine or interrupt service routine.

NOTE: When a CALL to subroutine or interrupt is acknowledged the Program Counter has already been incremented and is pointing to the 4-byte instruction group following the 4-byte group currently being executed. The instruction decoding logic allows the microprocessor to perform a test and execute a return conditional on the outcome of the test in a single cycle. A RETURN pops an address from the Return Stack and stores it to the Program Counter.

Return Instructions

Return-Always—Pop the top item from the Return Stack and transfer it to the Program Counter.

Return-If-Zero—If the TOP item on the Parameter Stack is "0" pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

return-If-Positive—If the most significant bit (sign bit) of the TOP item on the Parameter Stack is a "0" pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

Return-If-Carry-Clear—If the exponents of the floating point numbers found in TOP and NEXT are not aligned, pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

Return-Never—Execute the next instruction (NOP)

Return-If-Not-Zero—If the TOP item on the Parameter Stack is not "0" pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

Return-If-Negative—If the most significant bit (sign bit) of the TOP item on the Parameter Stack is a "1" pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

Return-If-Carry-Set—If the exponents of the floating point numbers found in TOP and NEXT are aligned, pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

HANDLING MEMORY FROM DYNAMIC RAM

The microprocessor **50**, like any RISC type architecture, is optimized to handle as many operations as possible on-chip for maximum speed. External memory operations take from 80 nsec. to 220 nsec. compared with on-chip memory speeds of from 4 nsec. to 30 nsec. There are times when external memory must be accessed.

External memory is accessed using three registers:

X-Register—A 30-bit memory pointer which can be used for memory access and simultaneously incremented or decremented.

Y-Register—A 30-bit memory pointer which can be used for memory access and simultaneously incremented or decremented.

Program-Counter—A 30-bit memory pointer normally used to point to 4-byte instruction groups. External memory may be accessed at addresses relative to the PC. The operands are sometimes called "Immediate" or "Literal" in other computers. When used as memory pointer, the PC is also incremented after each operation.

Memory Load & Store Instructions

Fetch-Via-X—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack. X is unchanged.

Fetch-Via-Y—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack. Y is unchanged.

Fetch-Via-X-Autoincrement—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack. After fetching, increment the most significant 30 bits of X to point to the next 32-bit word address.

Fetch-Via-Y-Autoincrement—Fetch the 32-bit memory content pointed to by Y and push it onto the Parameter Stack. After fetching, increment the most significant 30 bits of Y to point to the next 32-bit word address.

Fetch-Via-X-Autodecrement—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack. After fetching, decrement the most significant 30 bits of X to point to the previous 32-bit word address.

Fetch-Via-Y-Autodecrement—Fetch the 32-bit memory content pointed to by Y and push it onto the Parameter Stack. After fetching, decrement the most significant 30 bits of Y to point to the previous 32-bit word address.

Store-Via-X—Pop the top item of the Parameter Stack and store it in the memory location pointed to by X. X is unchanged.

Store-Via-X-Autoincrement—Pop the top item of the Parameter Stack and store it in the memory location pointed to by Y. Y is unchanged. STORE-VIA-X-AUTOINCREMENT - Pop the top item of the Parameter Stack and store it in the memory location pointed to by X. After storing, increment the most significant 30 bits of X to point to the next 32-bit word address.

Store-Via-Y-Autoincrement—Pop the top item of the Parameter Stack and store it in the memory location pointed to by Y. After storing, increment the most significant 30 bits of Y to point to the next 32-bit word address.

Store-Via-X-Autodecrement—Pop the top item of the Parameter Stack and store it in the memory location pointed to by X. After storing, decrement the most significant **30** bits of X to point to the previous 32-bit word address.

Store-Via-Y-Autodecrement—Pop the top item of the Parameter Stack and store it in the memory location pointed to by Y. After storing, decrement the most significant 30 bits of Y to point to the previous 32-bit word address.

Fetch-Via-PC—Fetch the 32-bit memory content pointed to by the Program Counter and push it onto the Parameter Stack. After fetching, increment the most significant 30 bits of the Program Counter to point to the next 32-bit word address.

*NOTE When this instruction executes, the PC is pointing to the memory location following the

instruction The effect is of loading a 32-bit immediate operand This is an 8-bit instruction and therefore will be combined with other 8-bit instructions in a 4-byte instruction fetch. It is possible to have from one to four FETCH-VIA-PC instructions in a 4-byte instruction fetch. The PC increments after each execution of FETCH-VIA-PC, so it is possible to push four immediate operands on the stack. The four operands would be the found in the four memory locations following the instruction.

Bye-Fetch-Via-X—Fetch the 32-bit memory content pointed to by the most significant 30 bits of X. Using the two least significant bits of X, select one of four bytes from the 32-bit memory fetch, right justify the byte in a 32-bit field and push the selected byte preceded by leading zeros onto the Parameter Stack

Byte-Store-Via-X—Fetch the 32-bit memory content pointed to by the most significant 30 bits of X Pop the TOP item from the Parameter Stack Using the two least significant bits of X place the least significant byte into the 32-bit memory data and write the 32-bit entity back to the location pointed to by the most significant 30 bits of X

Other Effects Of Memory Access Instructions

Any FETCH instruction will push a value on the Parameter Stack 74. If the on-chip stack is full, the stack will overflow into off-chip memory stack resulting in an additional memory cycle. Any STORE instruction will pop a value from the Parameter Stack 74. If the on-chip stack is empty, a memory cycle will be generated to fetch a value from off-chip memory stack.

Handling On-Chip Variables

High-level languages often allow the creation of LOCAL VARIABLES These variables are used by a particular procedure and discarded. In cases of nested procedures, layers of these variables must be maintained. On-chip storage is up to five times faster than off-chip RAM, so a means of keeping local variables on-chip can make operations run faster. The microprocessor 50 provides the capability for both on-chip storage of local variables and nesting of multiple levels of variables through the Return Stack.

The Return Stack 134 is implemented as 16 on-chip RAM locations. The most common use for the Return Stack 134 is storage of return addresses from subroutines and interrupt calls The microprocessor allows these 16 locations to also be used as addressable registers The 16 locations may be read and written by two instructions which indicate a Return Stack relative address from 0-15. When high-level procedures are nested, the current procedure variables push the previous procedure variables further down the Return Stack 134. Eventually, the Return Stack will automatically overflow into off-chip RAM

On-Chip Variable Instructions

Read-Local-Variable XXXX—Read the XXXXth location relative to the top of the Return Stack (XXXX is a binary number from 0000–1111). Push the item read onto the Parameter Stack. OTHER EFFECTS: If the Parameter Stack is full, the push operation will cause a memory cycle to be generated as one item of the stack is automatically stored to external RAM. The logic which selects the location performs a modulo 16 subtraction. If four local variables have been pushed onto the Return Stack, and an instruction attempts to READ the fifth item, unknown data will be returned

Write-Local-Variable XXXX—Pop the TOP item of the Parameter Stack and write it into the XXXXth location relative to the top of the Return Stack (XXXX is a binary number from 0000–1111.) OTHER EFFECTS: If the Parameter Stack is empty, the pop operation will cause a memory cycle to be generated to fetch the Parameter Stack item from external RAM The logic which selects the location performs a modulo 16 subtraction If four local variables have been pushed onto the Return Stack, and an instruction attempts to WRITE to the fifth item, it is possible to clobber return addresses or wreak other havoc

Register and Flip-Flop Transfer And Push Instructions

Drop—Pop the TOP item from the Parameter Stack and discard it.

Swap—Exchange the data in the TOP Parameter Stack location with the data in the NEXT Parameter Stack location

DUP—Duplicate the TOP item on the Parameter Stack and push it onto the Parameter Stack.

Push-Loop-CounterPush the value in LOOP COUNTER onto the Parameter Stack

Pop-RStack-Push-To-stack—Pop the top item from the Return Stack and push it onto the Parameter Stack.

Push-X-Reg—Push the value in the X Register onto the Parameter Stack.

Push-Stack-Pointer—Push the value of the Parameter Stack pointer onto the Parameter Stack.

Push-RStack-Pointer—Push the value of the Return Stack pointer onto the Return Stack.

Push-Mode-Bits—Push the value of the MODE REGISTER onto the Parameter Stack

Push-Input—Read the 10 dedicated input bits and push the value (right justified and padded with leading zeros) onto the Parameter Stack.

Set-Loop-Counter—Pop the TOP value from the Parameter Stack and store it into LOOP COUNTER.

Pop-Stack-Push-To-RStack—Pop the TOP item from the Parameter Stack and push it onto the Return Stack.

Set-X-Reg—Pop the TOP item from the Parameter Stack and store it into the X Register.

Set-Stack-Pointer—Pop the TOP item from the Parameter Stack and store it into the Stack Pointer

Set-RStack-Pointer—Pop the TOP item from the Parameter Stack and store it into the Return Stack Pointer.

Set-Mode-Bits—Pop the TOP value from the Parameter Stack and store it into the MODE BITS.

Set-output—Pop the TOP item from the Parameter Stack and output it to the 10 dedicated output bits OTHER EFFECTS: Instructions which push or pop the Parameter Stack or Return Stack may cause a memory cycle as the stacks overflow back and forth between on-chip and off-chip memory.

Loading A Short Literal

A special case of register transfer instruction is used to push an 8-bit literal onto the Parameter Stack This instruction requires that the 8-bits to be pushed reside in the last byte of a 4-byte instruction group. The instruc-

tion op-code loading the literal may reside in ANY of the other three bytes in the instruction group

Example

BYTE 1	BYTE 2	BYTE 3
LOAD-SHORT-LITERAL	QQQQQQQQ	QQQQQQQQ
BYTE 4		
00001111		

In this example, QQQQQQQQ indicates any other 8-bit instruction. When Byte 1 is executed, binary **00001111** (HEX Of) from Byte **4** will be pushed (right justified and padded by leading zeros) onto the Parameter Stack. Then the instructions in Byte 2 and Byte 3 will execute. The microprocessor instruction decoder knows not to execute Byte **4**. It is possible to push three identical 8-bit values as follows:

BYTE 1	BYTE 2
LOAD-SHORT-LITERAL	LOAD-SHORT-LITERAL
BYTE 3	BYTE 4
LOAD-SHORT-LITERAL	00001111

Short-literal-Instruction

Load-Short-Literal—Push the 8bit value found in Byte **4** of the current 4-byte instruction group onto the Parameter Stack

Logic Instructions

Logical and math operations used the stack for the source of one or two operands and as the destination for results. The stack organization is a particularly convenient arrangement for evaluating expressions. TOP indicates the top value on the Parameter Stack **74**. NEXT indicates the next to top value on the Parameter Stack **74**.

AND—Pop TOP and NEXT from the Parameter Stack, perform the logical AND operation on these two operands, and push the result onto the Parameter Stack.

OR—Pop TOP and NEXT from the Parameter Stack, perform the logical OR operation on these two operands, and push the result onto the Parameter Stack

XOR—Pop TOP and NEXT from the Parameter Stack, perform the logical exclusive OR on these two operands, and push the result onto the Parameter Stack.

Bit-Clear—Pop TOP and NEXT from the Parameter Stack, toggle all bits in NEXT, perform the logical AND operation on TOP, and push the result onto the Parameter Stack. (Another way of understanding this instruction is thinking of it as clearing all bits in TOP that are set in NEXT.)

Math Instructions

Math instruction pop the TOP item and NEXT to top item of the Parameter Stack **74** to use as the operands. The results are pushed back on the Parameter Stack. The CARRY flag is used to latch the "33rd bit" of the ALU result.

Add—Pop the TOP item and NEXT to top item from the Parameter Stack, add the values together and push the result back on the Parameter Stack. The CARRY flag may be changed.

Add-With-Carry—Pop the TOP item and the NEXT to top item from the Parameter Stack, add the values together. If the CARRY flag is "1" increment the result. Push the ultimate result back on the Parameter Stack. The CARRY flag may be changed.

ADD-X—Pop the TOP item from the Parameter Stack and read the third item from the top of the Parameter Stack. Add the values together and push the result back on the Parameter Stack. The CARRY flag may be changed.

SUB—Pop the TOP item and NEXT to top item from the Parameter Stack, Subtract NEXT from TOP and push the result back on the Parameter Stack. The CARRY flag may be changed.

SUB-WITH-CARRY—Pop the TOP item and NEXT to top item from the Parameter Stack. Subtract NEXT from TOP. If the CARRY flag is "1" increment the result. Push the ultimate result back on the Parameter Stack. The CARRY flag may be changed.

SUB-X—

SIGNED-MULT-STEP—

UNSIGNED-MULT-STEP—

SIGNED-FAST-MULT—

FAST-MULT-STEP—

UNSIGNED-DIV-STEP—

GENERATE-POLYNOMIAL—

ROUND—

COMPARE—Pop the TOP item and NEXT to top item from the Parameter Stack. Subtract NEXT from TOP. If the result has the most significant bit equal to "0" (the result is positive), push the result onto the Parameter Stack. If the result has the most significant bit equal to "1" (the result is negative), push the old value of TOP onto the Parameter Stack. The CARRY flag may be affected.

Shift/Rotate

SHIFT-LEFT—Shift the TOP Parameter Stack item left one bit. The CARRY flag is shifted into the least significant bit of TOP.

SHIFT-RIGHT—Shift the TOP Parameter Stack item right one bit. The least significant bit of TOP is shifted into the CARRY flag. Zero is shifted into the most significant bit of TOP.

DOUBLE-SHIFT-LEFT—Treating the TOP item of the Parameter Stack as the most significant word of a 64-bit number and the NEXT stack item as the least significant word, shift the combined 64-bit entity left one bit. The CARRY flag is shifted into the least significant bit of NEXT.

DOUBLE-SHIFT-RIGHT—Treating the TOP item of the Parameter Stack as the most significant word of a 64-bit number and the NEXT stack item as the least significant word, shift the combined 64-bit entity right one bit. The least significant bit of NEXT is shifted into the CARRY flag. Zero is shifted into the most significant bit of TOP.

Other Instructions

FLUSH-STACK—Empty all on-chip Parameter Stack locations into off-chip RAM. (This instruction useful for multitasking applications). This instruction accesses a counter which holds the depth

of the on-chip stack and can require from none to 16 external memory cycles.

FLUSH-RSTACK—Empty all on-chip Return Stack locations into off-chip RAM. (This instruction is useful for multitasking applications). This instruction accesses a counter which holds the depth of the on-chip Return Stack and can require from none to **16** external memory cycles

It should further be apparent to those skilled in the art that various changes in form and details of the invention as shown and described may be made It is intended that such changes be included within the spirit and scope of the claims appended hereto.

What is claimed is:

1. A microprocessor system, comprising a central processing unit integrated circuit, a memory extend of said central processing unit integrated circuit, a bus connecting said central processing unit integrated circuit to said memory, and means connected to said bus for fetching instructions for said central processing unit integrated circuit on said bus from said memory, said means for fetching instructions being configured and connected to fetch multiple sequential instructions from said memory in parallel and supply the multiple sequential instructions to said central processing unit integrated circuit during a single memory cycle, said bus having a width at least equal to a number of bits in each of the instructions times a number of the instructions fetched in parallel, said central processing unit including an arithmetic logic unit and a first push down stack connected to said arithmetic, logic unit, said first push down stack including means for storing a top item connected to a first input of said arithmetic logic unit to provide the top item to the first input and means for storing a next item connected to a second input of said arithmetic logic unit to provide the next item to the second input, a remainder of said first push down stack being connected to said means for storing a next item to receive the next item from said means for storing a next item when pushed down in said push down stack said arithmetic logic unit having an output connected to said means for storing a top item.

2. The microprocessor system of claim **1** additionally comprising means connected to said means for fetching multiple instructions for determining by decoding the multiple instructions if multiple instructions fetched by said means for fetching multiple instructions require a memory access, said means for fetching multiple instructions fetching additional multiple instructions if decoding the multiple instructions shows that the multiple instructions do not require a memory access.

3 The microprocessor system of claim **2** in which the decoding determines if the multiple instructions do not require a memory access by a state of a bit of each of the multiple instructions -

4. The microprocessor system of claim **3** in which the bit is a most significant bit of the multiple instructions

5. The microprocessor system of claim **1** additionally comprising an instruction register for the multiple instructions connected to said means for fetching instructions, means connected to said instruction register for supplying the multiple instructions in succession from said instruction register, a counter connected to control said means for supplying the multiple instructions to supply the multiple instructions in succession, means for decoding the multiple instructions connected to receive the multiple instructions in succession from the means for supplying the multiple instructions, said counter being connected to said means for decoding to receive incrementing and reset control signals from said means for decoding, said means for decoding being configured to supply the reset control signal to said counter and to supply a control signal to said means for fetching instructions in response to a SKIP instruction in the multiple instructions

6. The microprocessor system of claim **5** additionally comprising a loop counter connected to receive a decrement control signal from said means for decoding, said means for decoding being configured to supply the reset control signal to said counter and the decrement control signal to said loop counter in response to a MICROLOOP instruction in the multiple instructions to provide a microloop within the multiple instructions in said instruction register for a number of repetitions controlled by said loop counter.

7. The microprocessor system of claim **1** additionally comprising an instruction register for the multiple instructions and a variable width operand to be used with one of the multiple instructions connected to said means for fetching instructions, means connected to said instruction register for supplying the multiple instructions in succession from said instruction register, a counter connected to control said means for supplying the multiple instructions to supply the multiple instructions in succession,

means for decoding the multiple instructions connected to receive the multiple instructions in succession from the means for supplying the multiple instructions, said counter being connected to said means for decoding to receive incrementing and reset control signals from said means for decoding, said means for decoding being configured to control said counter in response to an instruction utilizing the variable width operand stored in said instruction register, and means connected to said counter to select the variable width operand for use with the instruction utilizing the variable width operand in response to said counter.

8. A microprocessor system, comprising a central processing unit, a memory, a bus connecting said central processing unit to said memory, and means connected to said bus for fetching instructions for said central processing unit on said bus from said memory, said means for fetching instructions being configured and connected to fetch multiple sequential instructions from said memory in parallel and supply the multiple sequential instructions to said central processing unit during a single memory cycle. said central processing unit including an arithmetic logic unit and a first push down stack connected to said arithmetic logic unit, said first push down stack further including means for storing a top item connected to a first input of said arithmetic logic unit to provide the top item to the first input, means for storing a next item connected to a second input of said arithmetic logic unit to provide the next item to the second input, said arithmetic logic unit having an output connected to said means for storing a top item, a second push down stack, said means for storing a top item being connected to provide an input to said second push down stack and a control means connected between said means for storing a top item and said second push down stack for controlling provision of the input to said second push down stack, said second push down stack additionally being configured as a register file and said means for storing a top item and said sec-

ond push down stack additionally configured as the register file being bidirectionally connected.

9. A microprocessor system, comprising a central processing unit, a dynamic random access memory, a bus connecting said central processing unit to said dynamic random access memory, and multiplexing means on said bus between said central processing unit and said dynamic random access memory, said multiplexing means being connected and configured to provide multiplexed row addresses, column addresses and data on said bus from said central processing unit to said dynamic random access memory and to provide data from said dynamic random access memory to said central processing unit, and

means connected to said bus for fetching instructions for said central processing unit on said bus from said dynamic random access memory, said means for fetching instructions being configured to fetch multiple sequential instructions from said dynamic random access memory in parallel and supply the multiple instructions to said central processing unit during a single memory cycle,

said central processing unit including an arithmetic logic unit and a first push down stack connected to said arithmetic logic unit, said first push down stack including means for storing a top item connected to a first input of said arithmetic logic unit to provide the top item to the first input, and means for storing a next item connected to a second input of said arithmetic logic unit to provide the next item to the second input, a remainder of said first push down stack being connected to said means for storing a next item to receive the next item from said means for storing a next item when pushed down in said push down stack, said arithmetic logic unit having an output connected to said means for storing a top item.

10. The microprocessor system of claim 9 additionally comprising a second push down stack said means for storing a top item being connected to provide an input to said second push down stack and a control means connected between said means for storing a top item and said second push down stack for controlling provision of the input to said second push down stack.

11. The microprocessor system of claim 10 in which said second push down stack is additionally configured as a register file and said means for storing a top item and said second push down stack additionally configured as the register file are bidirectionally connected.

12. The microprocessor system of claim 11 additionally comprising means connected to said means for fetching multiple instructions for determining by decoding the multiple instructions if multiple instructions fetched by said means for fetching multiple instructions require a memory access, said means for fetching multiple instructions fetching additional multiple instructions if decoding the multiple instructions shows that the multiple instructions do not require a memory access.

13. The microprocessor system of claim 12 additionally comprising an instruction register for the multiple instructions connected to said means for fetching instructions, means connected to said instruction register for supplying the multiple instructions in succession from said instruction register, a counter connected to control said means for supplying the multiple instructions to supply the multiple instructions in succession, means for decoding the multiple instructions connected to receive the multiple instructions in succession from the means for supplying the multiple instructions, said counter being connected to said means for decoding to receive incrementing and reset control signals from said means for decoding, said means for decoding being configured to supply the reset control signal to said counter and to supply a control signal to said means for fetching instructions in response to a SKIP instruction in the multiple instructions.

14. The microprocessor system of claim 13 additionally comprising a loop counter connected to receive a decrement control signal from said means for decoding, said means for decoding being configured to supply the reset control signal to said counter and the decrement control signal to said loop counter in response to a MICROLOOP instruction in the multiple instructions within the multiple instructions in said instruction register for a number of repetitions controlled by said loop counter.

15. The microprocessor system of claim 13 in which said means for decoding is configured to control said counter in response to one of the multiple instructions utilizing a variable width operand stored in said instruction register with the multiple instructions, said microprocessor system additionally comprising means connected to said counter to select the variable width operand for use with the instruction utilizing the variable width operand in response to a state of said counter resulting from control of said counter by said means for decoding.

16. The microprocessor system of claim 12 in which the decoding determines if the multiple instructions do not require a memory access by a state of a bit of each of the multiple instructions.

17. The microprocessor system of claim 16 in which the bit is a most significant bit of the multiple instructions.

18. The microprocessor system of claim 9 additionally comprising a programmable read only memory containing instructions connected to said bus, means connected to said bus for fetching instructions for said central processing unit on said bus, said means for fetching instructions including means for assembling a plurality of instructions from said programmable read only memory, storing the plurality of instructions in said dynamic random access memory and subsequently supplying the plurality of instructions from said dynamic random access memory to said central processing unit on said bus.

19. The microprocessor system of claim 9 additionally comprising a direct memory access processing unit having the capacity to request and execute instructions, said bus connecting said direct memory access processing unit to said dynamic random access memory, said dynamic random access memory containing instructions for said central processing unit and said direct memory access processing unit, said direct memory access processing unit being connected to means for fetching instructions for said central processing unit on said bus and for fetching instructions for said direct memory access processing unit on said bus.

20. The microprocessor system of claim 19 additionally comprising a variable speed system clock connected to said central processing unit and a fixed speed system clock connected to control said means for fetching instructions for said central processing unit and for fetching instructions for said direct memory access processing unit.

21. The microprocessor system of claim 9 in which said microprocessor system is configured to provide

different memory access timing for different storing capacity sizes of said dynamic random access memory by including a sensing circuit and a driver circuit, and an output enable line connected between said dynamic random access memory, said sensing circuit and said driver circuit, said sensing circuit being configured to provide a ready signal when said output enable line reaches a predetermined electrical level after a memory read operation as a function of different capacitance on said bus as a result of the different storing capacity sizes of said dynamic random access memory, said microprocessor system being configured so that said driver circuit provides an enabling signal on said output enable line responsive to the ready signal

22. The microprocessor system of claim 21 in which the predetermined electrical level is a predetermined voltage.

23. The microprocessor system of claim 9 in which said microprocessor system is configured to operate at a variable clock speed; said microprocessor system additionally comprising a ring counter variable speed system clock connected to said central processing unit, said central processing unit and said ring counter variable speed system clock being provided in a single integrated circuit said ring counter variable speed system clock being configured to provide different clock speed to said central processing unit as a result of transistor propagation delays, depending on at least one of temperature of said single integrated circuit, voltage and microprocessor fabrication process for said single integrated circuit.

24. The microprocessor system of claim 23 additionally comprising an input/output interface connected between said microprocessor system and an external memory bus to exchange coupling control signals, addresses and data between said central processing unit and said input/output interface, and a second clock independent of said ring counter variable speed system clock connected t said input/output interface to provide clock signals for operation of said input/output interface asynchronously from said central processing unit.

25. The microprocessor system of claim 24 in which said second clock is a fixed frequency clock.

26. The microprocessor system of claim 9 in which said first push down stack has a first plurality of stack registers having stack memory elements configured as latches, a second plurality of stack registers having stack memory elements configured as a random access memory, said first and second plurality of stack registers and said central processing unit being provided in a single integrated circuit with a top one of said second plurality of stack registers being connected to said a bottom one of said first plurality of stack registers, and a third plurality of stack registers having stack memory elements configured as a random access memory external to said single integrated circuit, with a top one of said third plurality of stack registers being connected to a bottom one of said second plurality of stack registers, said microprocessor system being configured to operate said first, second and third plurality of stack registers hierarchically as interconnected stacks.

27. The microprocessor system of claim 26 additionally comprising a first pointer connected to said first plurality of stack registers, a second pointer connected to said second plurality of stack registers, and a third pointer connected to said third plurality of stack registers, said microprocessor system being configured to operate said first, second and third plurality of stack registers hierarchically as interconnected stacks by having said central processing unit being connected to pop items from said first plurality of stack registers, said first stack pointer being connected to said second stack pointer to pop a first plurality of items from said second plurality of stack registers when said first plurality of stack registers are empty from successive pop operations by said central processing unit, said second stack pointer being connected to said third stack pointer to pop a second plurality of items from said third plurality of stack registers when said second plurality of stack registers are empty from successive pop operations by said central processing unit.

28. The microprocessor system of claim 9 additionally comprising a first register connected to supply a first input to said arithmetic logic unit, a first shifter connected between an output of said arithmetic logic unit and said first register, a second register connected to receive a starting polynomial value, an output of said second register being connected to a second shifter, a least significant bit of said second register being connected to said arithmetic logic unit, a third register connected to supply feedback terms of a polynomial to said arithmetic logic unit, a down counter, for counting down a number corresponding to digits of a polynomial to be generated, connected to said arithmetic logic unit, said arithmetic logic unit being responsive to a polynomial instruction to carry out an exclusive OR of the contents of said first register with the contents of said third register if the least significant bit of said second register is a "ONE" and to pass the contents of said first register unaltered if the least significant bit of said second register is a "ZERO" until said down counter completes a count, the polynomial to be generated resulting in said first register.

29. The microprocessor system of claim 28 in which said first register is a result register, said first shifter is a left shifting shifter, said second register is a multiplier register connected to receive a multiplier in bit reversed form, said second shifter is a right shifting shifter, said third register is connected to supply a multiplicand to said arithmetic logic unit, said down counter is configured for counting down a number corresponding to one less than the number of digits of the multiplier, said arithmetic logic unit being responsive to a multiply instruction to add the contents of said result register with the contents of said third register, if the least significant bit of said second register is a "ONE" and to pass the contents of said first register unaltered if the least significant bit of said second register is a "ZERO" until said down counter completes a count, the product resulting in said first register.

* * * * *

US006598148B1

(12) United States Patent
Moore et al.

(10) Patent No.: US 6,598,148 B1
(45) Date of Patent: Jul. 22, 2003

(54) HIGH PERFORMANCE MICROPROCESSOR HAVING VARIABLE SPEED SYSTEM CLOCK

(75) Inventors: **Charles H. Moore**, Woodside, CA (US); **Russell H. Fish, III**, Dallas, TX (US)

(73) Assignee: **Patriot Scientific Corporation**, Poway, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/124,623**

(22) Filed: **Jul. 29, 1998**

(Under 37 CFR 1.47)

Related U.S. Application Data

(62) Division of application No. 08/484,918, filed on Jun. 7, 1995, now Pat. No. 5,809,336, which is a division of application No. 07/389,334, filed on Aug. 3, 1989, now Pat. No. 5,440,749.

(51) **Int. Cl.**[7] **G06F 15/00**
(52) **U.S. Cl.** **712/32**
(58) **Field of Search** 712/32; 711/104, 711/105

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,680,698 A	*	7/1987	Edwards et al.	712/37
5,379,438 A	*	1/1995	Bell et al.	712/37

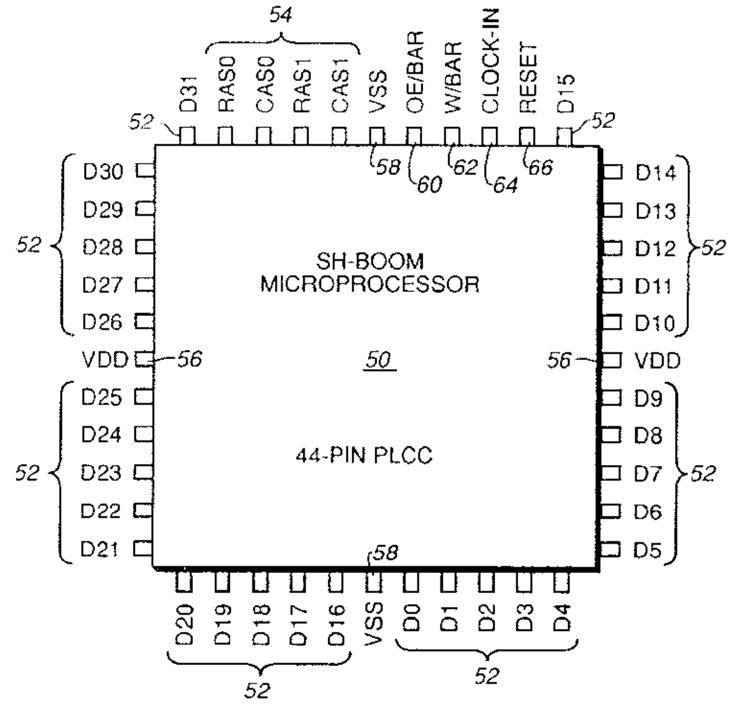
* cited by examiner

Primary Examiner—David Y. Eng
(74) *Attorney, Agent, or Firm*—Knobbe Martens Olson & Bear LLP

(57) **ABSTRACT**

A microprocessor integrated circuit including a processing unit disposed upon an integrated circuit substrate is disclosed herein. The processing unit is designed to operate in accordance with a predefined sequence of program instructions stored within an instruction register. A memory capable of storing information provided by the processing unit and occupying a larger area of the integrated circuit substrate than the processing unit, is also provided within the microprocessor integrated circuit. The memory may be implemented using, for example dynamic or static random-access memory. A variable output frequency system clock, such as generated by a ring oscillator, is also disposed on the integrated circuit substrate.

13 Claims, 19 Drawing Sheets



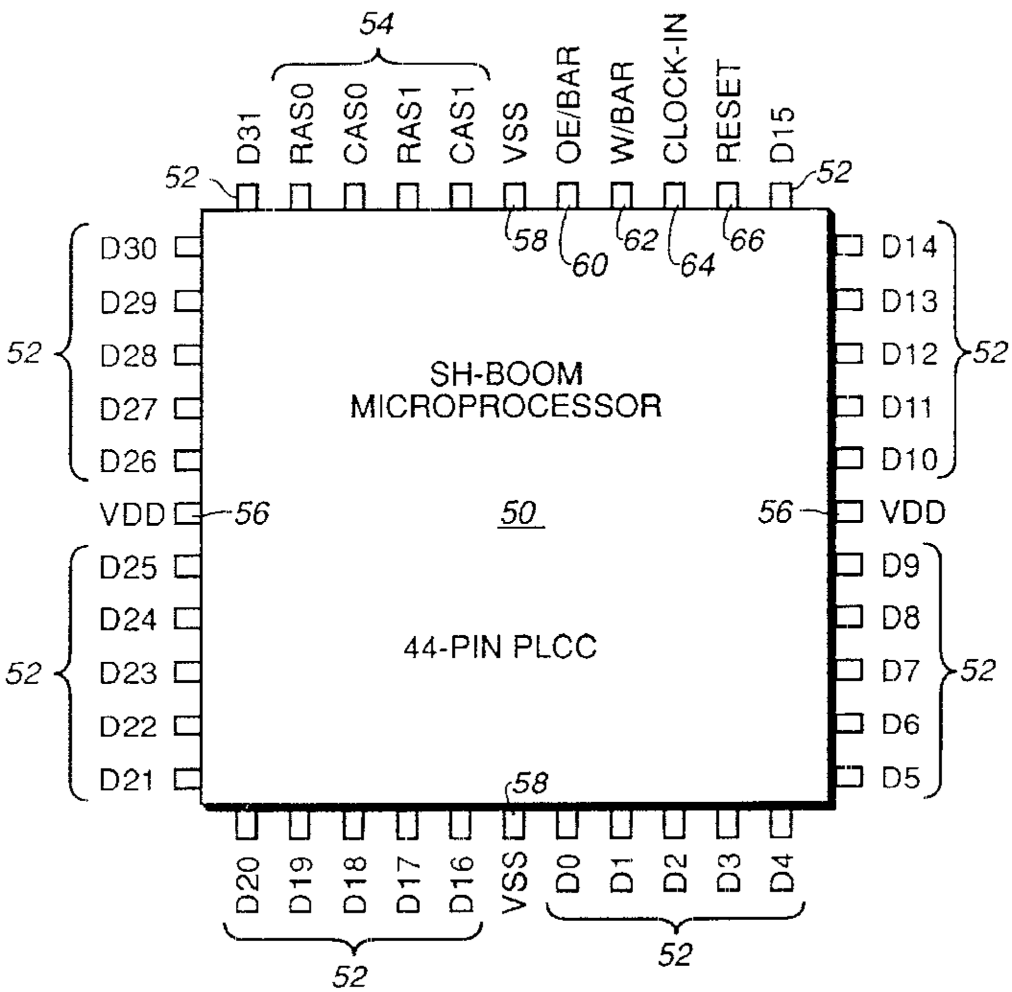

54
D31
RAS0
CAS0
RAS1
CAS1
VSS
OE/BAR
W/BAR
CLOCK-IN
RESET
D15
52
58
60
62
64
66
D30
D29
D28
D27
D26
VDD
56
D25
D24
D23
D22
D21
D14
D13
D12
D11
D10
VDD
D9
D8
D7
D6
D5
SH-BOOM
MICROPROCESSOR
50
44-PIN PLCC
D20
D19
D18
D17
D16
VSS
D0
D1
D2
D3
D4

FIG._1

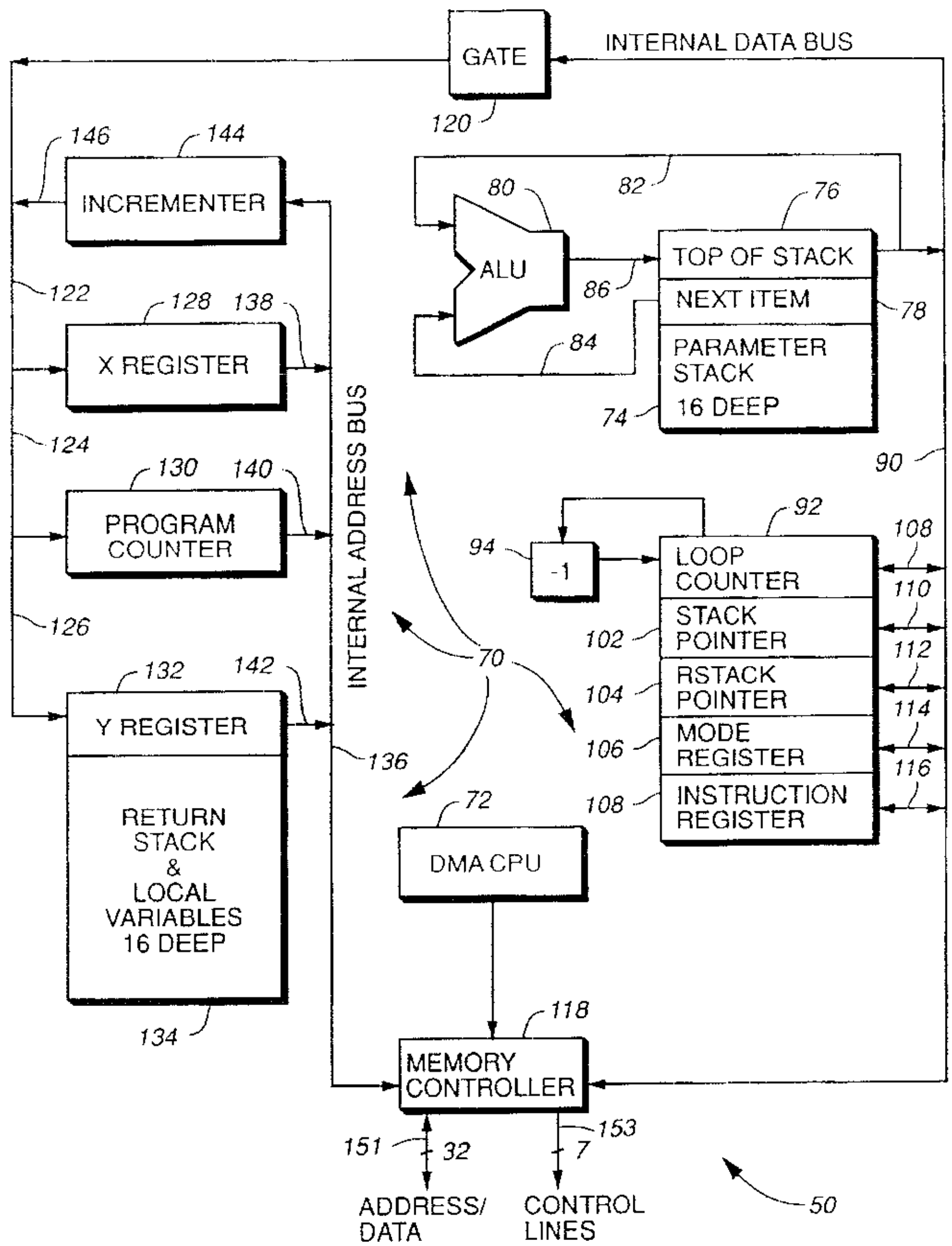
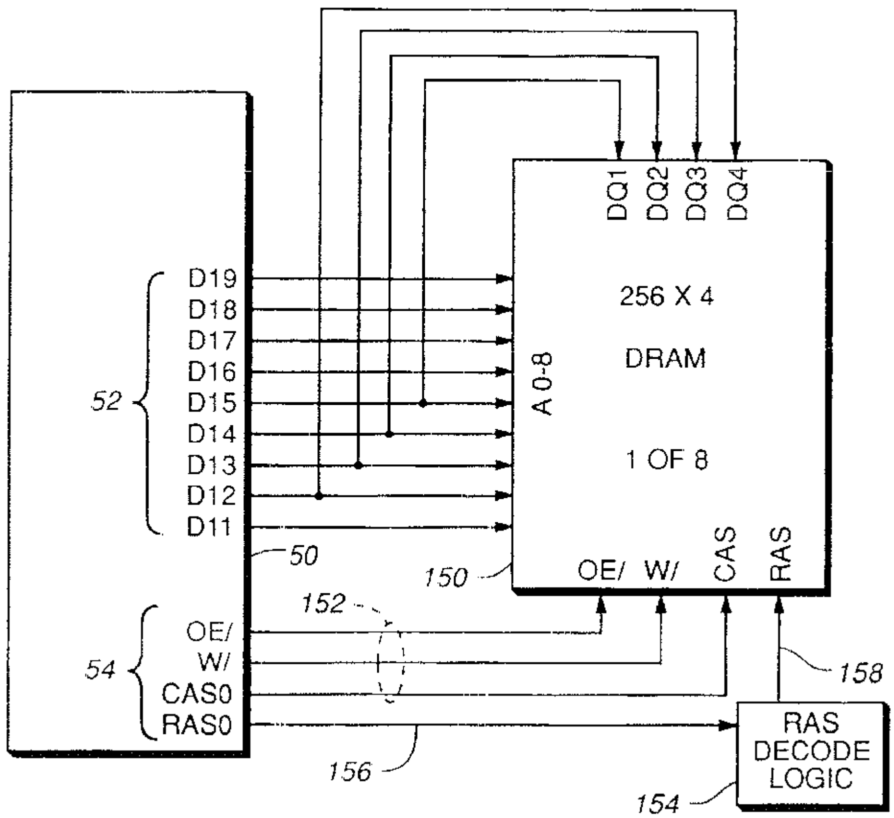
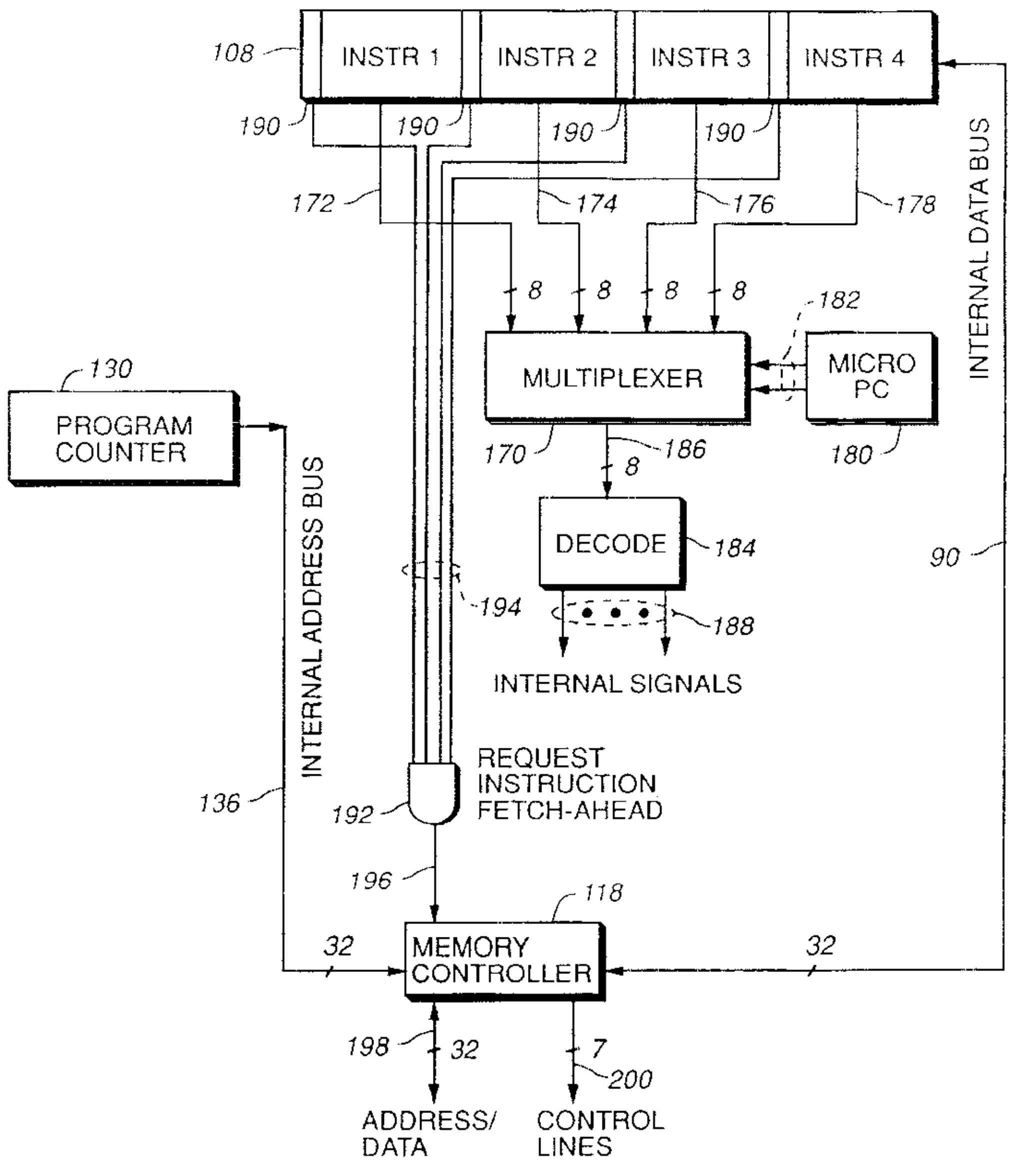

GATE
INTERNAL DATA BUS
120
146
144
INCREMENTER
122
128
138
X REGISTER
124
130
140
PROGRAM COUNTER
126
132
142
Y REGISTER
RETURN STACK & LOCAL VARIABLES 16 DEEP
134
136
INTERNAL ADDRESS BUS
80
82
76
ALU
86
84
TOP OF STACK
NEXT ITEM
PARAMETER STACK 16 DEEP
74
78
90
92
94
-1
LOOP COUNTER
STACK POINTER
RSTACK POINTER
MODE REGISTER
INSTRUCTION REGISTER
102
104
106
108
108
110
112
114
116
70
72
DMA CPU
118
MEMORY CONTROLLER
151
32
153
7
ADDRESS/ DATA
CONTROL LINES
50

FIG._2



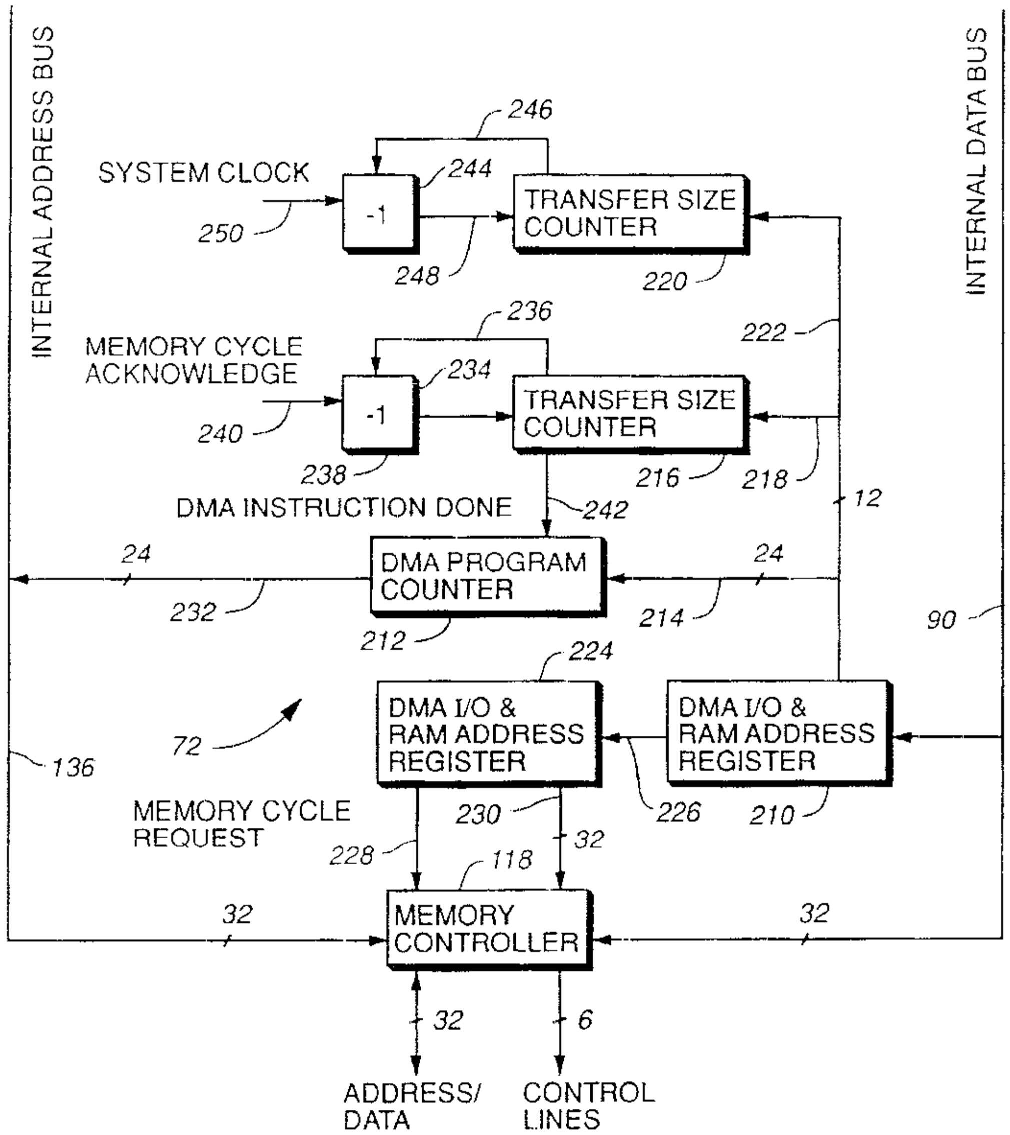
D19
D18
D17
D16
D15
D14
D13
D12
D11
52
50
DQ1
DQ2
DQ3
DQ4
256 X 4
DRAM
A 0-8
1 OF 8
150
OE/
W/
CAS
RAS
152
54
OE/
W/
CAS0
RAS0
158
RAS DECODE LOGIC
156
154

FIG._3



108
INSTR 1
INSTR 2
INSTR 3
INSTR 4
190
190
190
190
172
174
176
178
INTERNAL DATA BUS
8
8
8
8
182
MULTIPLEXER
MICRO PC
130
PROGRAM COUNTER
170
186
180
8
INTERNAL ADDRESS BUS
DECODE
184
90
194
188
INTERNAL SIGNALS
REQUEST INSTRUCTION FETCH-AHEAD
136
192
196
118
32
MEMORY CONTROLLER
32
198
32
7
200
ADDRESS/ DATA
CONTROL LINES

FIG._4



INTERNAL ADDRESS BUS
INTERNAL DATA BUS
SYSTEM CLOCK
250
-1
248
244
246
TRANSFER SIZE COUNTER
220
222
MEMORY CYCLE ACKNOWLEDGE
240
-1
238
234
236
TRANSFER SIZE COUNTER
216
218
12
DMA INSTRUCTION DONE
242
24
DMA PROGRAM COUNTER
232
212
24
214
90
224
DMA I/O & RAM ADDRESS REGISTER
DMA I/O & RAM ADDRESS REGISTER
72
136
226
210
MEMORY CYCLE REQUEST
230
32
228
118
32
MEMORY CONTROLLER
32
32
6
ADDRESS/ DATA
CONTROL LINES

FIG._5

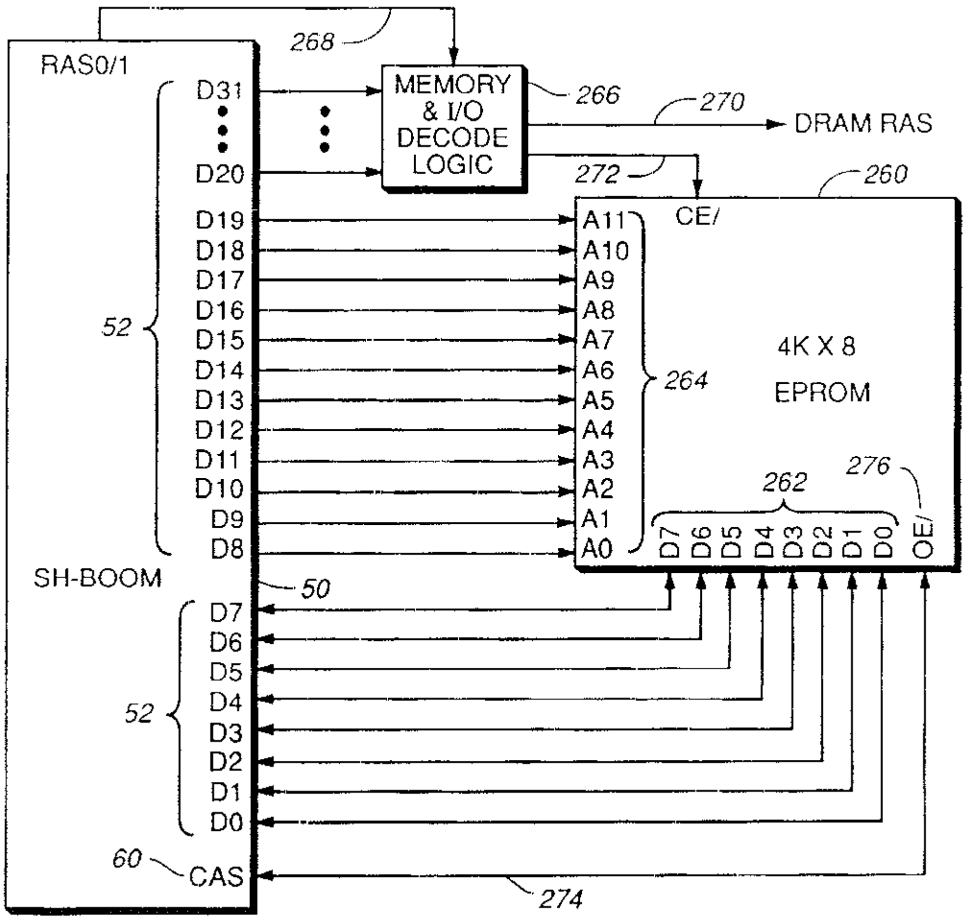
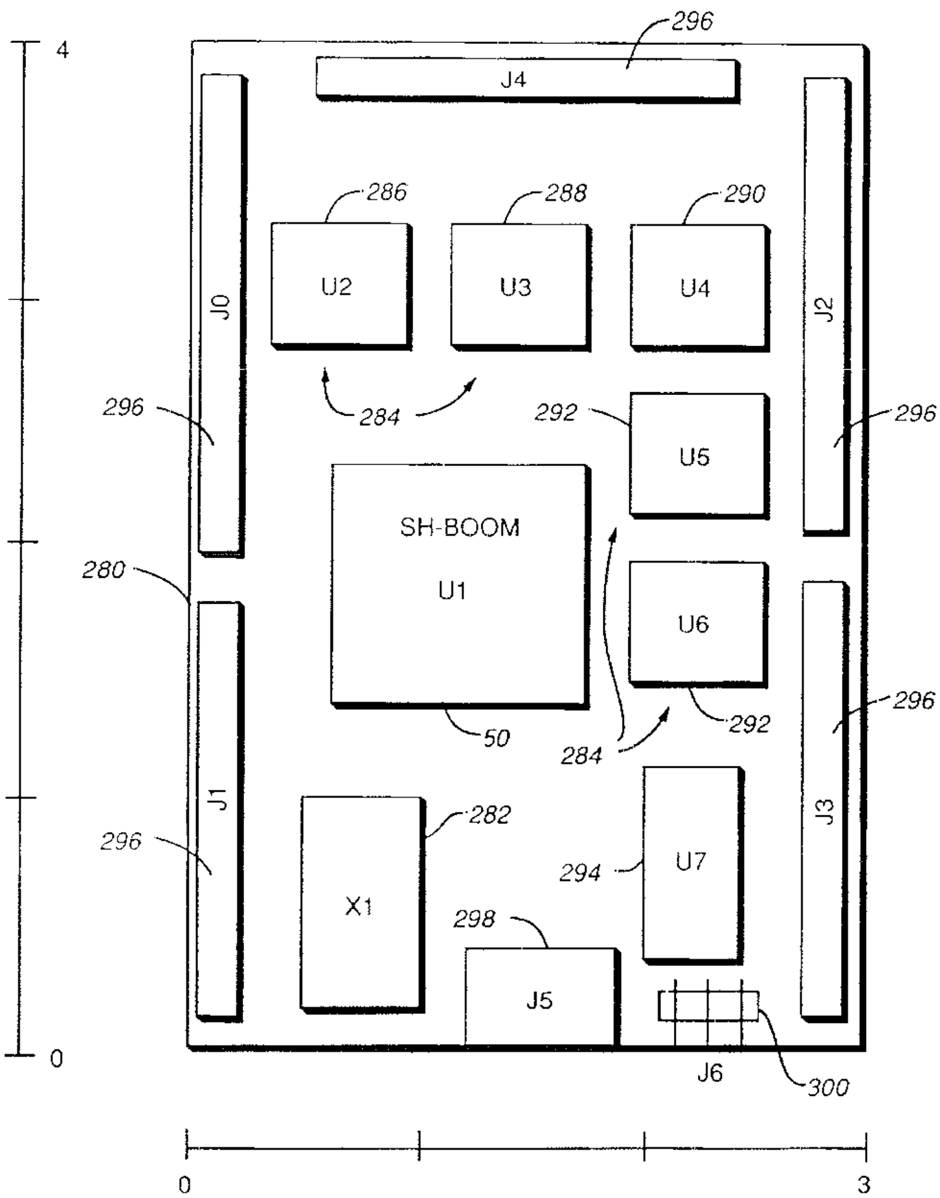

RAS0/1
268
MEMORY & I/O DECODE LOGIC
266
270
DRAM RAS
272
260
D31
D20
D19
D18
D17
D16
D15
D14
D13
D12
D11
D10
D9
D8
52
CE/
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
264
4K X 8
EPROM
262
276
D7
D6
D5
D4
D3
D2
D1
D0
OE/
SH-BOOM
50
D7
D6
D5
D4
D3
D2
D1
D0
52
60
CAS
274

FIG._6



4
0
296
J4
J0
286
U2
288
U3
290
U4
J2
296
284
292
U5
296
SH-BOOM
U1
280
U6
296
50
284
292
J1
J3
296
X1
282
294
U7
298
J5
J6
300
0
3

FIG._7

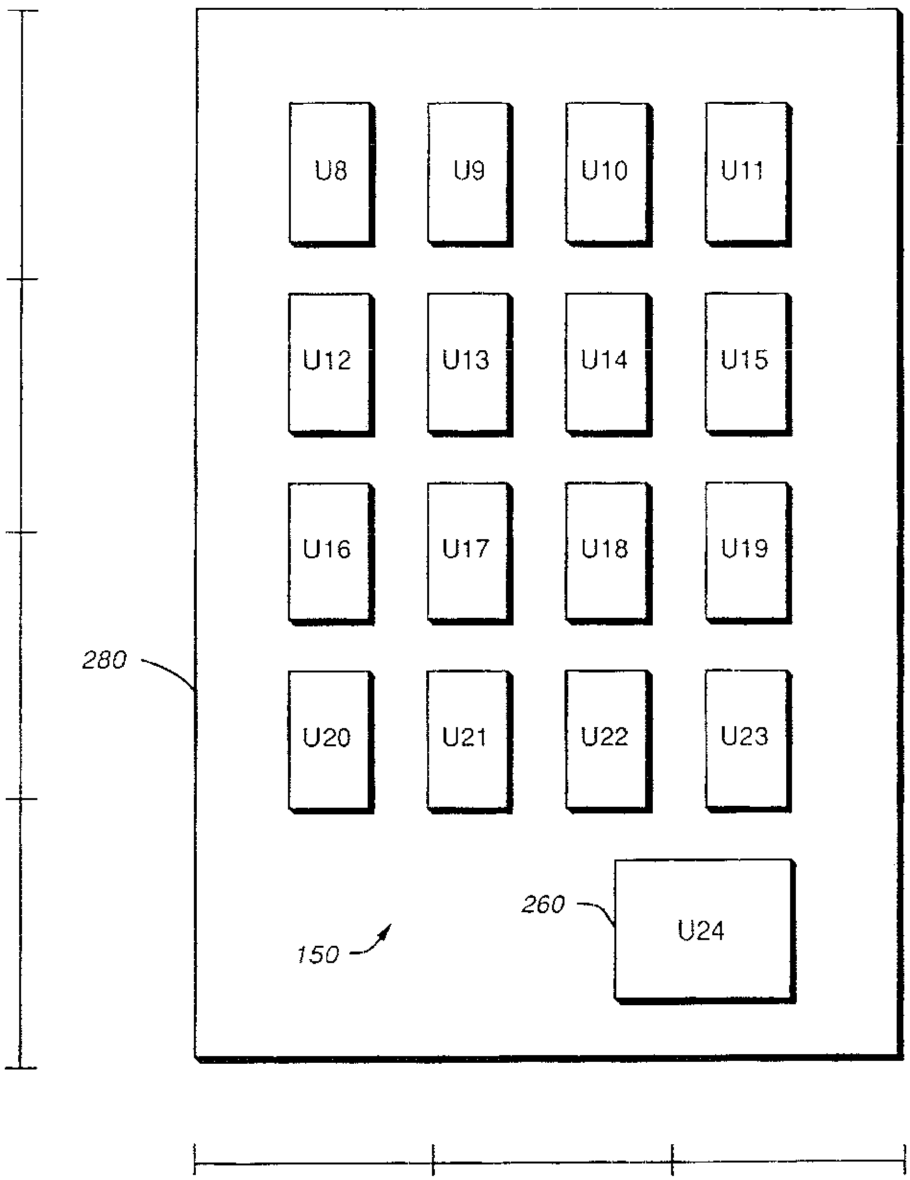

U8
U9
U10
U11
U12
U13
U14
U15
U16
U17
U18
U19
U20
U21
U22
U23
U24
280
260
150

FIG._8

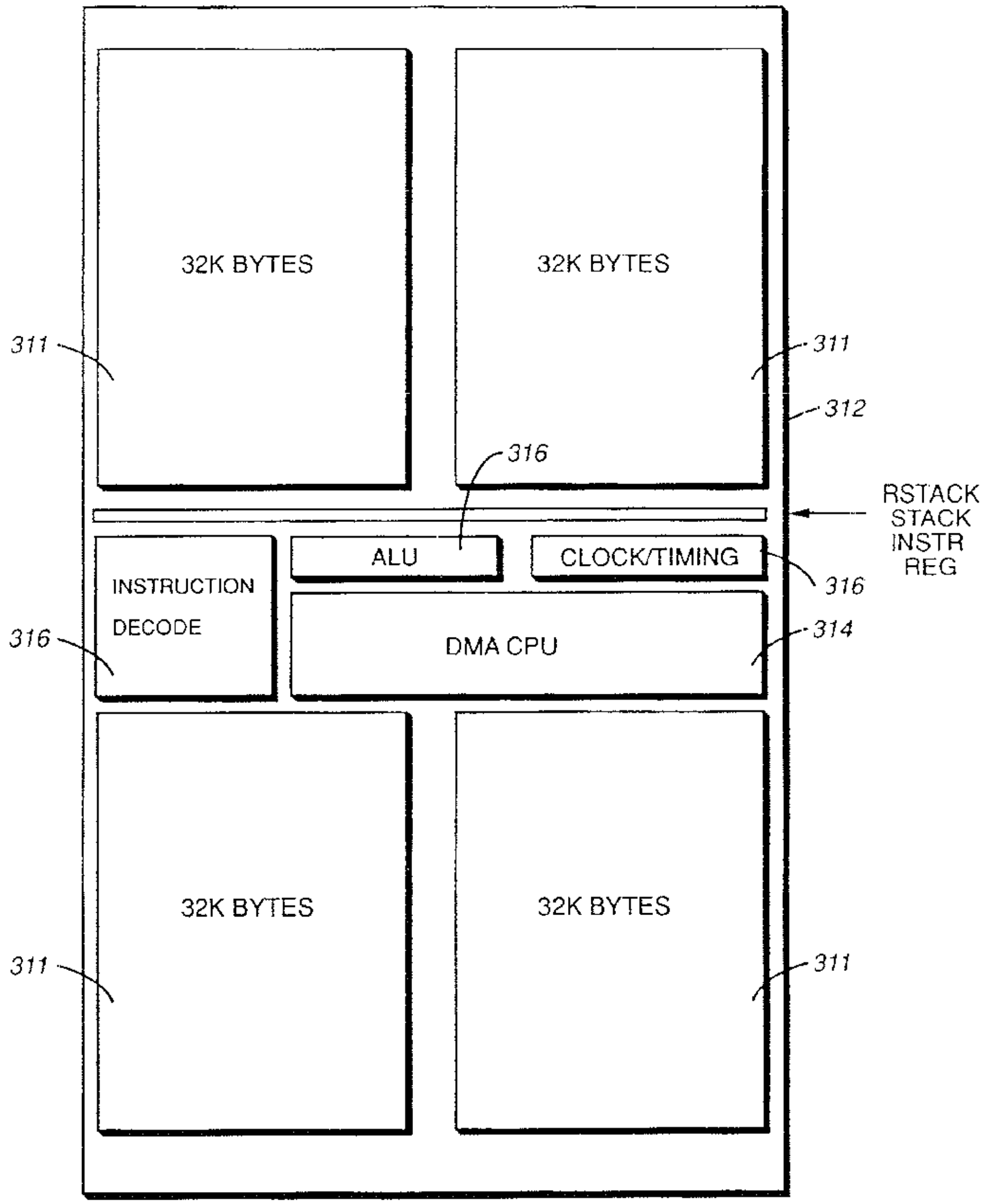

32K BYTES
32K BYTES
311
311
312
316
RSTACK
STACK
INSTR
REG
ALU
CLOCK/TIMING
INSTRUCTION
DECODE
316
316
DMA CPU
314
32K BYTES
32K BYTES
311
311

FIG._9

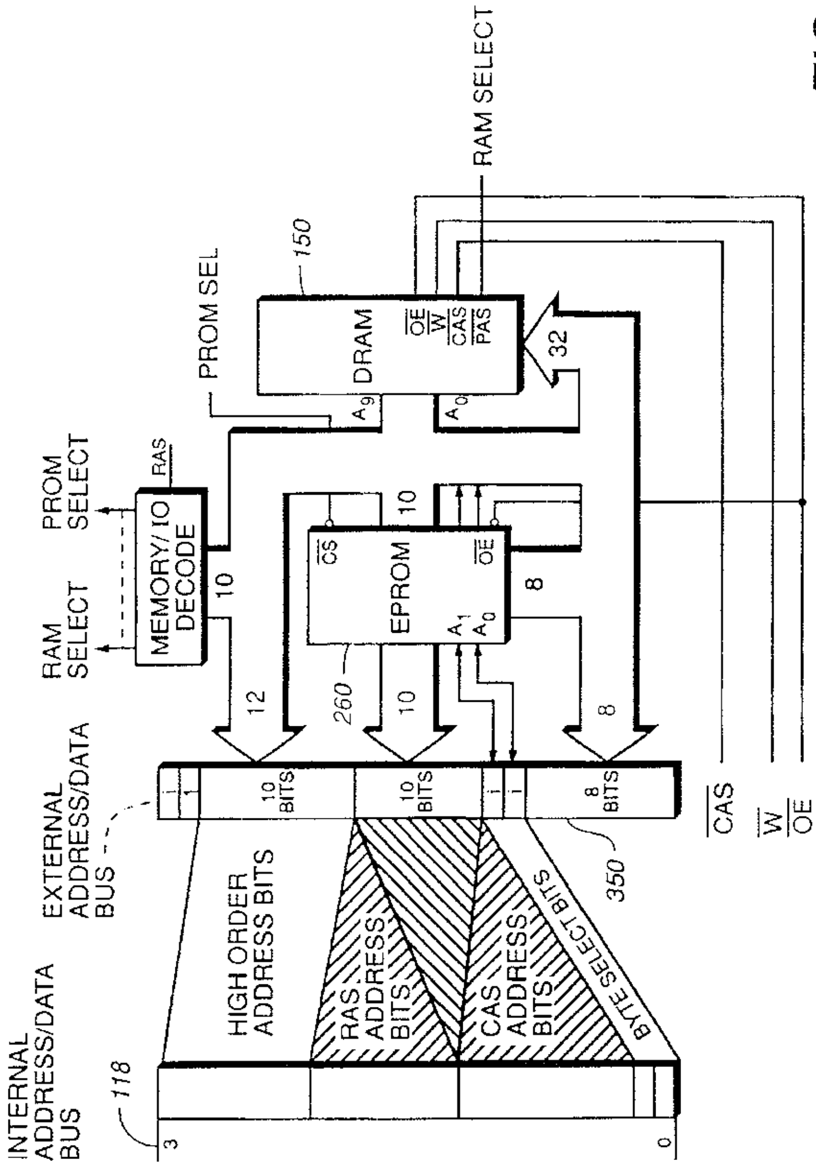
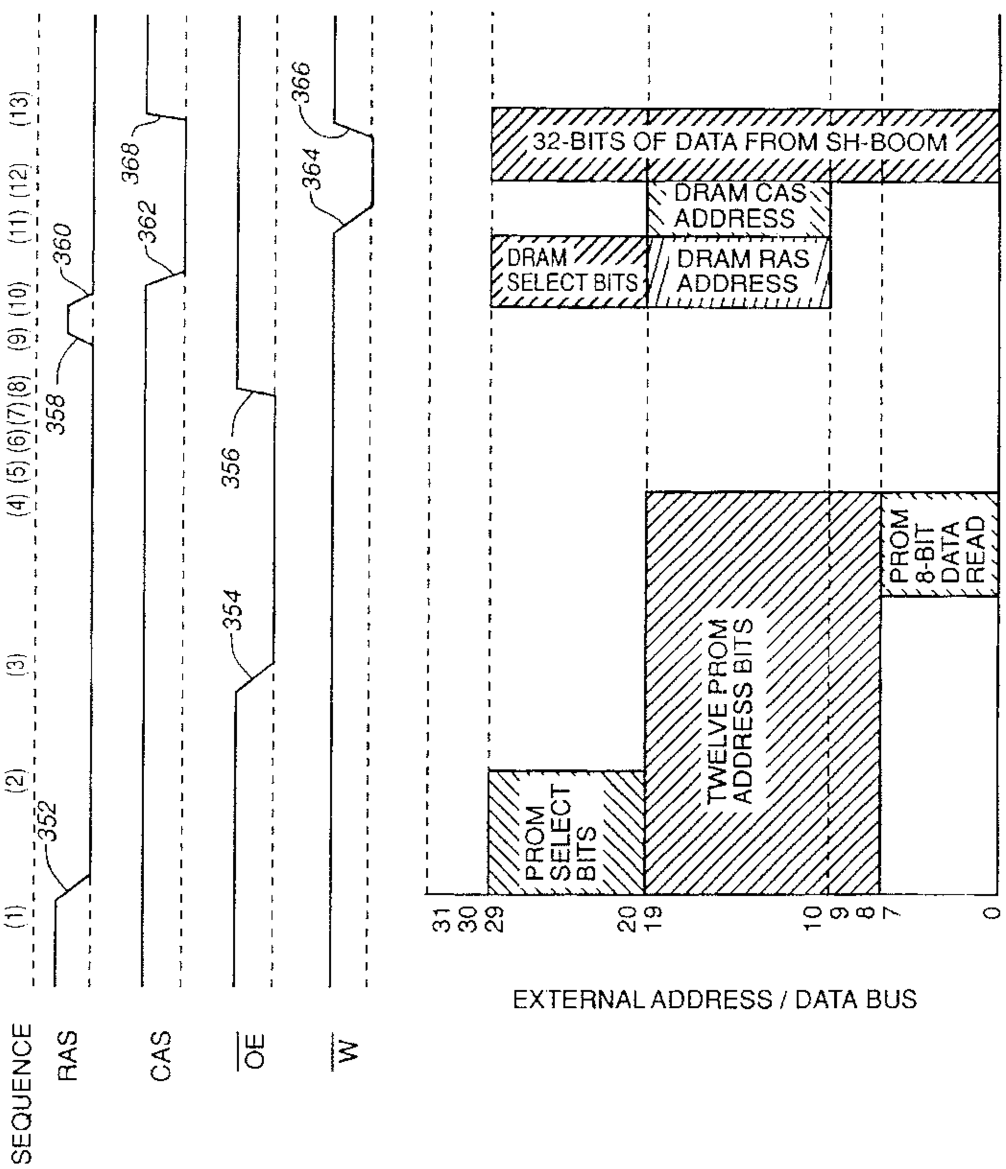

RAM SELECT
PROM SEL
150
DRAM
OE
W
CAS
PAS
A_9
A_0
32
PROM SELECT
RAM SELECT
RAS
MEMORY/ IO DECODE
10
EPROM
CS
OE
A_1
A_0
10
8
260
12
10
8
EXTERNAL ADDRESS/DATA BUS
10 BITS
10 BITS
8 BITS
CAS
W
OE
350
HIGH ORDER ADDRESS BITS
RAS ADDRESS BITS
CAS ADDRESS BITS
BYTE SELECT BITS
INTERNAL ADDRESS/DATA BUS
118
3
0

FIG._10

FIG._11


SEQUENCE
RAS
CAS
OE
W
(1)
(2)
(3)
(4) (5) (6) (7) (8)
(9) (10)
(11) (12)
(13)
352
354
356
358
360
362
364
366
368
32-BITS OF DATA FROM SH-BOOM
DRAM CAS ADDRESS
DRAM SELECT BITS
DRAM RAS ADDRESS
PROM SELECT BITS
TWELVE PROM ADDRESS BITS
PROM 8-BIT DATA READ
31
30
29
20
19
10
9
8
7
0
EXTERNAL ADDRESS / DATA BUS

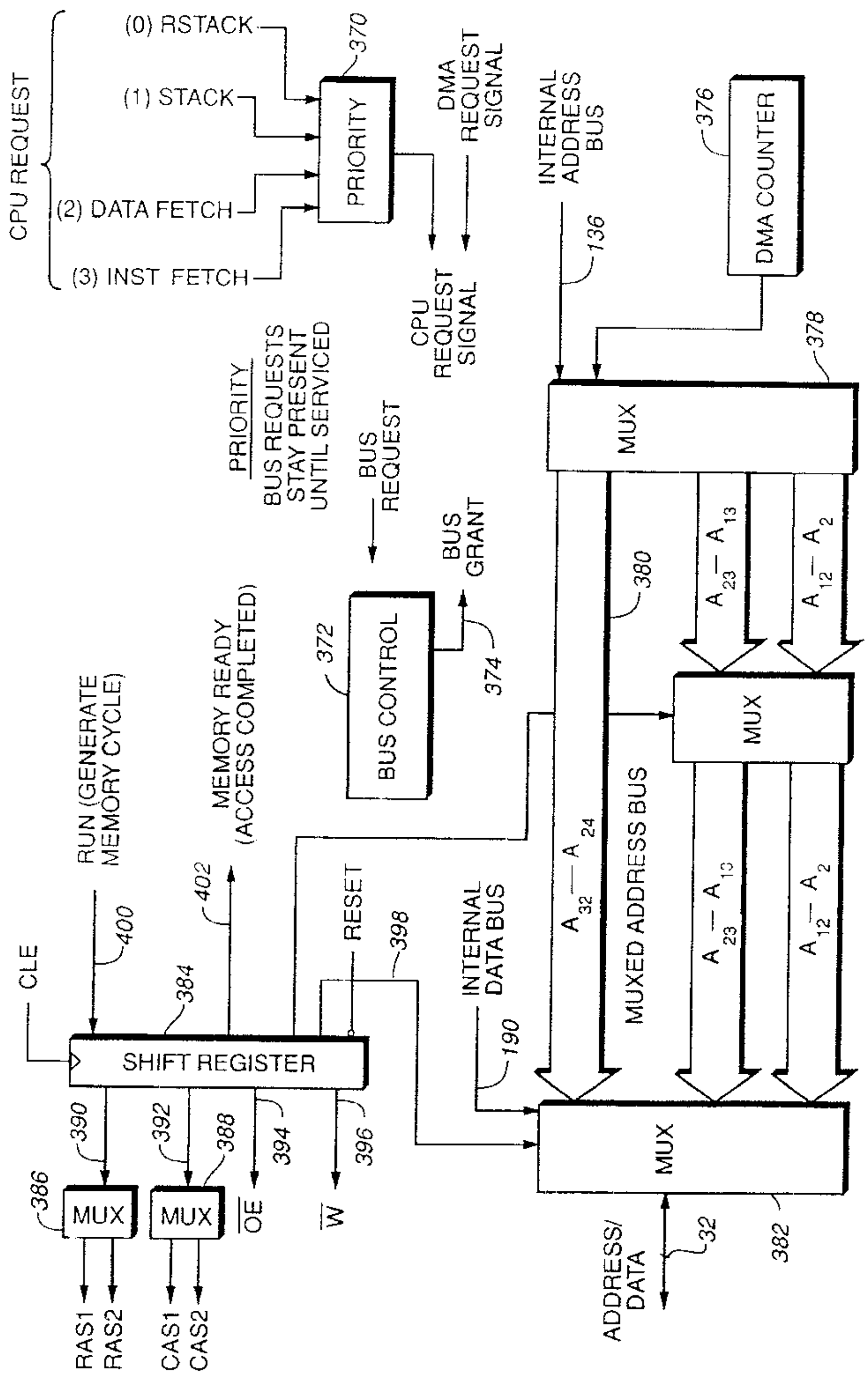
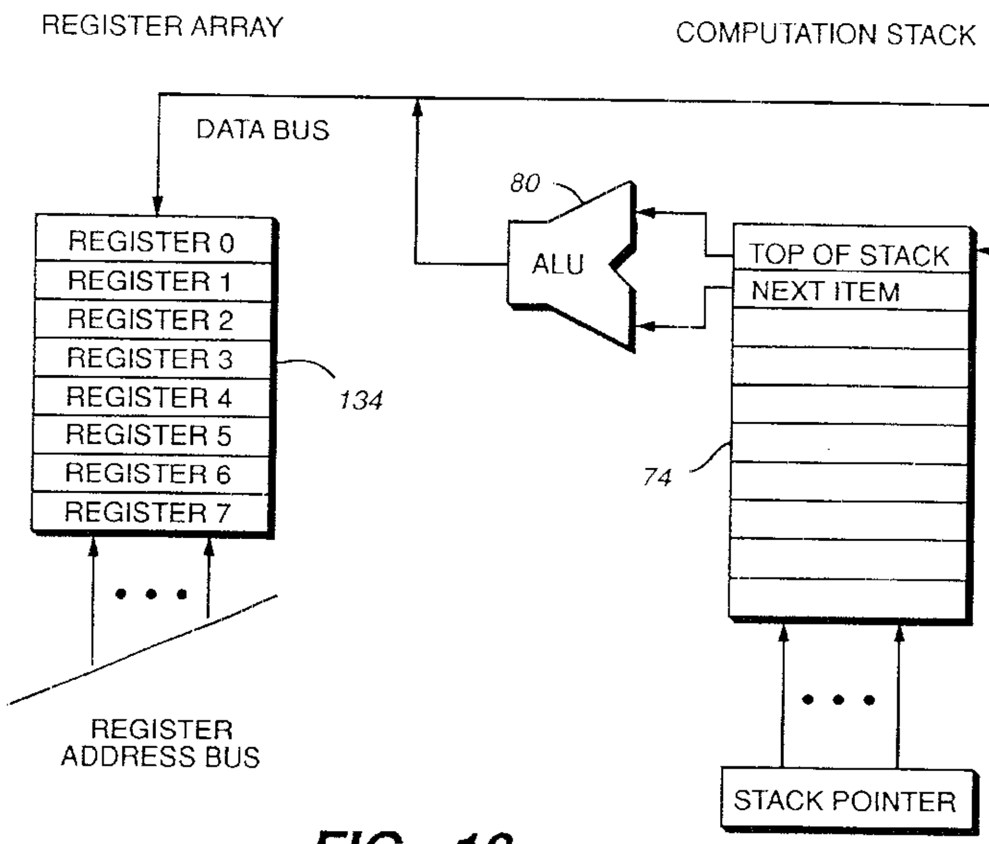

CPU REQUEST
(0) RSTACK
(1) STACK
(2) DATA FETCH
(3) INST FETCH
PRIORITY
370
DMA REQUEST SIGNAL
CPU REQUEST SIGNAL
INTERNAL ADDRESS BUS
136
376
DMA COUNTER
378
MUX
PRIORITY
BUS REQUESTS STAY PRESENT UNTIL SERVICED
BUS REQUEST
BUS GRANT
372
BUS CONTROL
374
380
A_{23} — A_{13}
A_{12} — A_2
MUX
MUXED ADDRESS BUS
A_{32} — A_{24}
A_{23} — A_{13}
A_{12} — A_2
RUN (GENERATE MEMORY CYCLE)
MEMORY READY (ACCESS COMPLETED)
400
402
RESET
398
INTERNAL DATA BUS
190
CLE
384
SHIFT REGISTER
390
392
388
394
396
386
MUX
MUX
$\overline{OE}$
$\overline{W}$
RAS1
RAS2
CAS1
CAS2
MUX
ADDRESS/ DATA
32
382

FIG._12



REGISTER ARRAY
COMPUTATION STACK
DATA BUS
80
ALU
TOP OF STACK
NEXT ITEM
REGISTER 0
REGISTER 1
REGISTER 2
REGISTER 3
REGISTER 4
REGISTER 5
REGISTER 6
REGISTER 7
134
74
REGISTER ADDRESS BUS
STACK POINTER

FIG._13

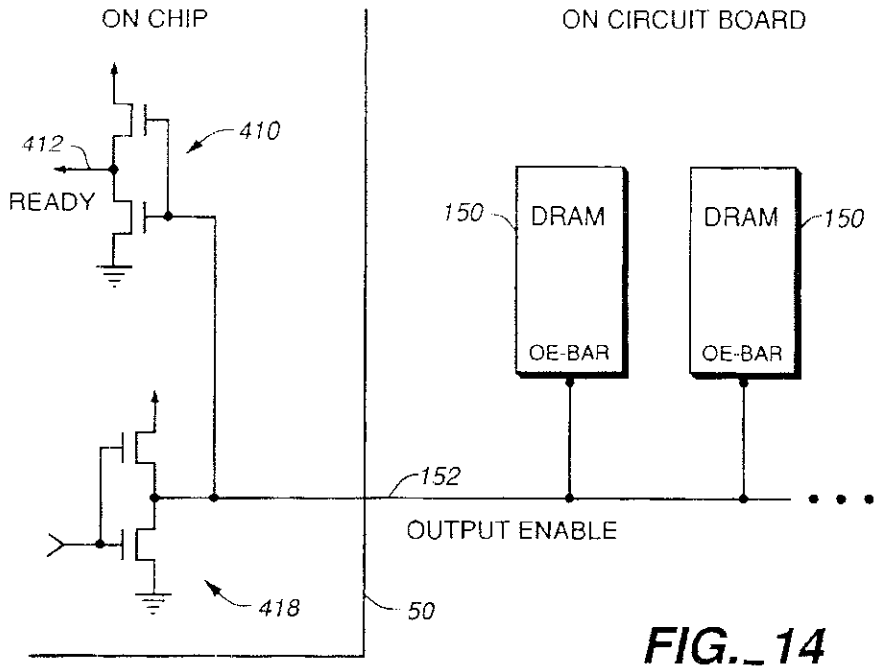

ON CHIP
ON CIRCUIT BOARD
412
410
READY
150
DRAM
OE-BAR
DRAM
OE-BAR
150
152
OUTPUT ENABLE
418
50

FIG._14

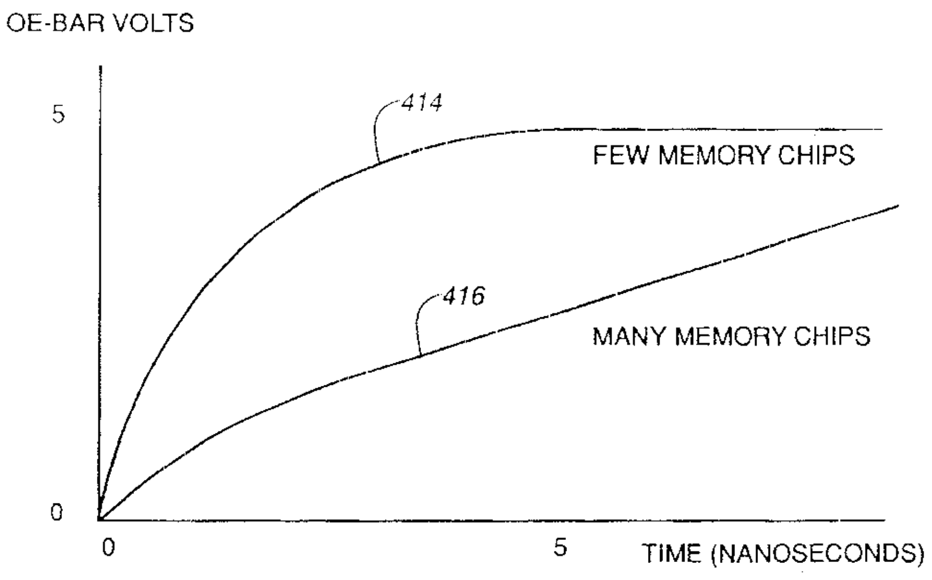

OE-BAR VOLTS
5
414
FEW MEMORY CHIPS
416
MANY MEMORY CHIPS
0
0
5
TIME (NANOSECONDS)

FIG._15

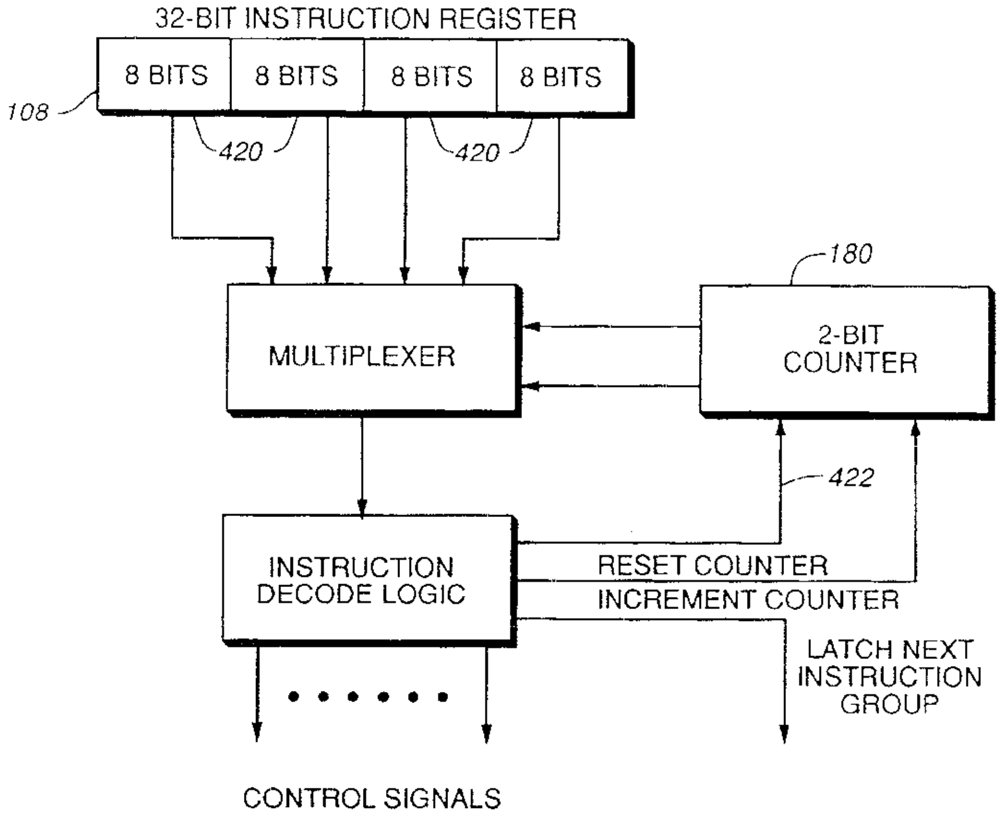

32-BIT INSTRUCTION REGISTER
8 BITS
8 BITS
8 BITS
8 BITS
108
420
420
180
MULTIPLEXER
2-BIT COUNTER
422
INSTRUCTION DECODE LOGIC
RESET COUNTER
INCREMENT COUNTER
LATCH NEXT INSTRUCTION GROUP
CONTROL SIGNALS

FIG._16

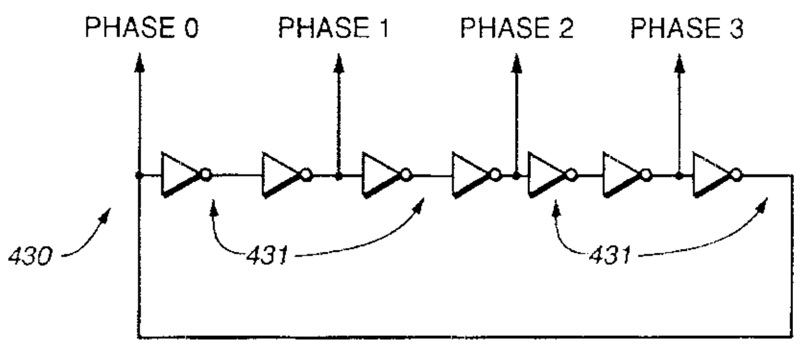
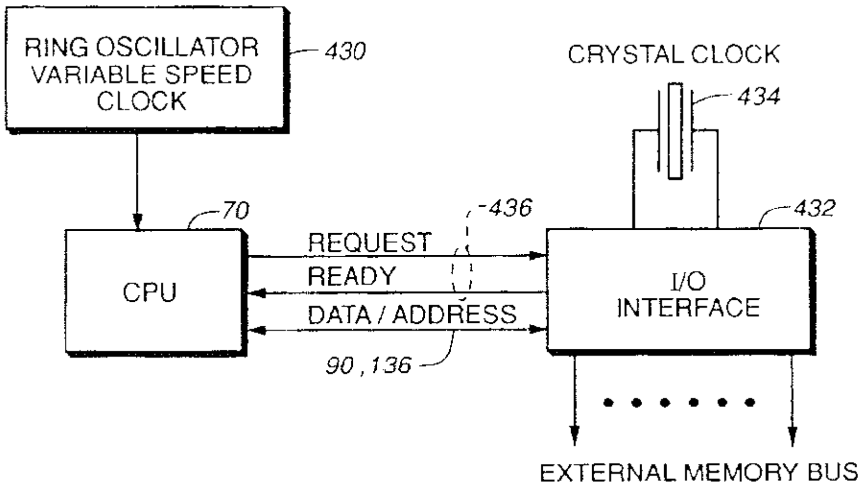
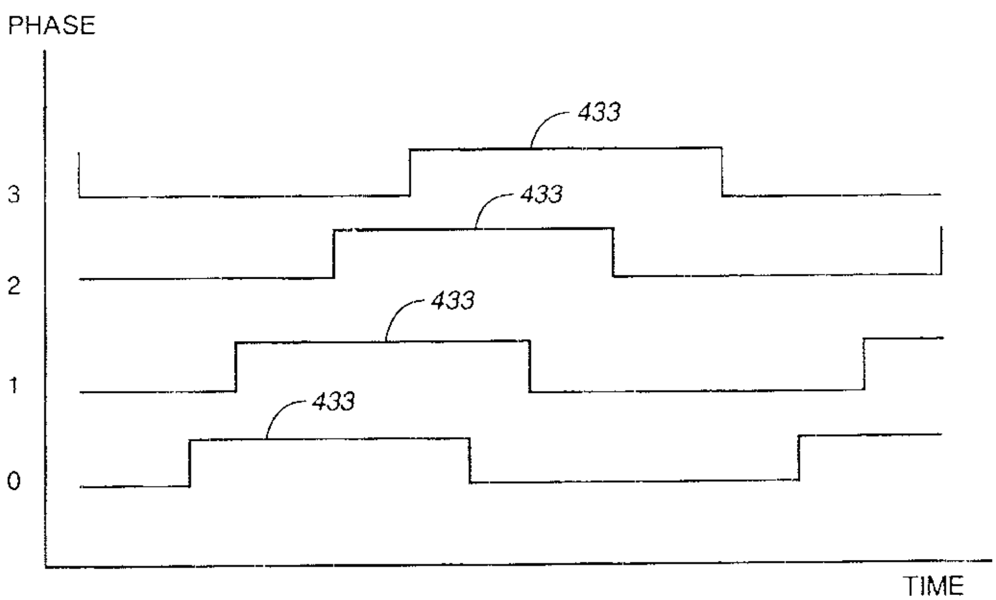

PHASE 0
PHASE 1
PHASE 2
PHASE 3
430
431
431

FIG._18



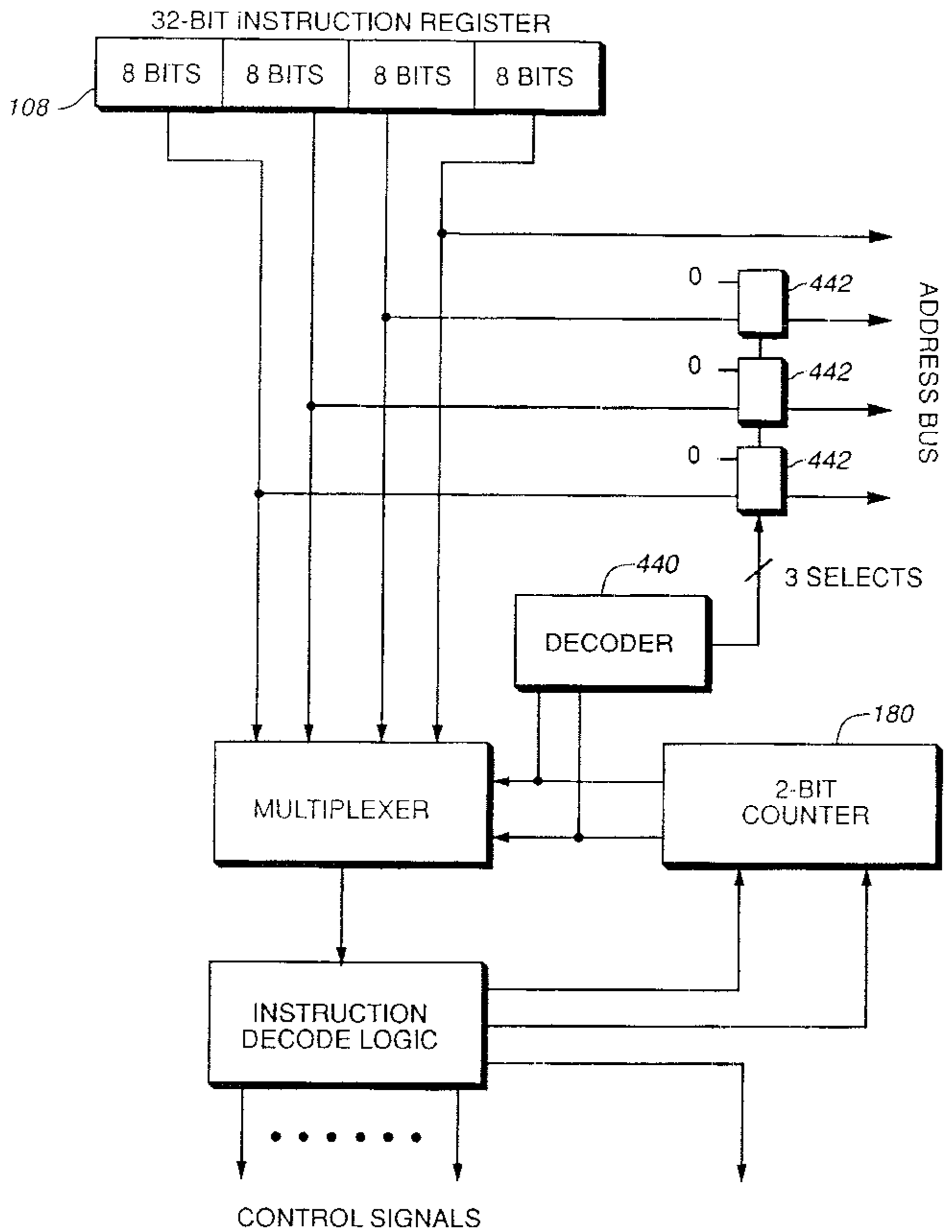
RING OSCILLATOR VARIABLE SPEED CLOCK
430
CRYSTAL CLOCK
434
70
436
432
REQUEST
READY
DATA / ADDRESS
CPU
I/O INTERFACE
90 , 136
EXTERNAL MEMORY BUS

FIG._17



PHASE
433
3
2
1
0
TIME

FIG._19



32-BIT INSTRUCTION REGISTER
8 BITS
8 BITS
8 BITS
8 BITS
108
0
442
0
442
0
442
ADDRESS BUS
440
3 SELECTS
DECODER
180
MULTIPLEXER
2-BIT
COUNTER
INSTRUCTION
DECODE LOGIC
CONTROL SIGNALS

FIG._20

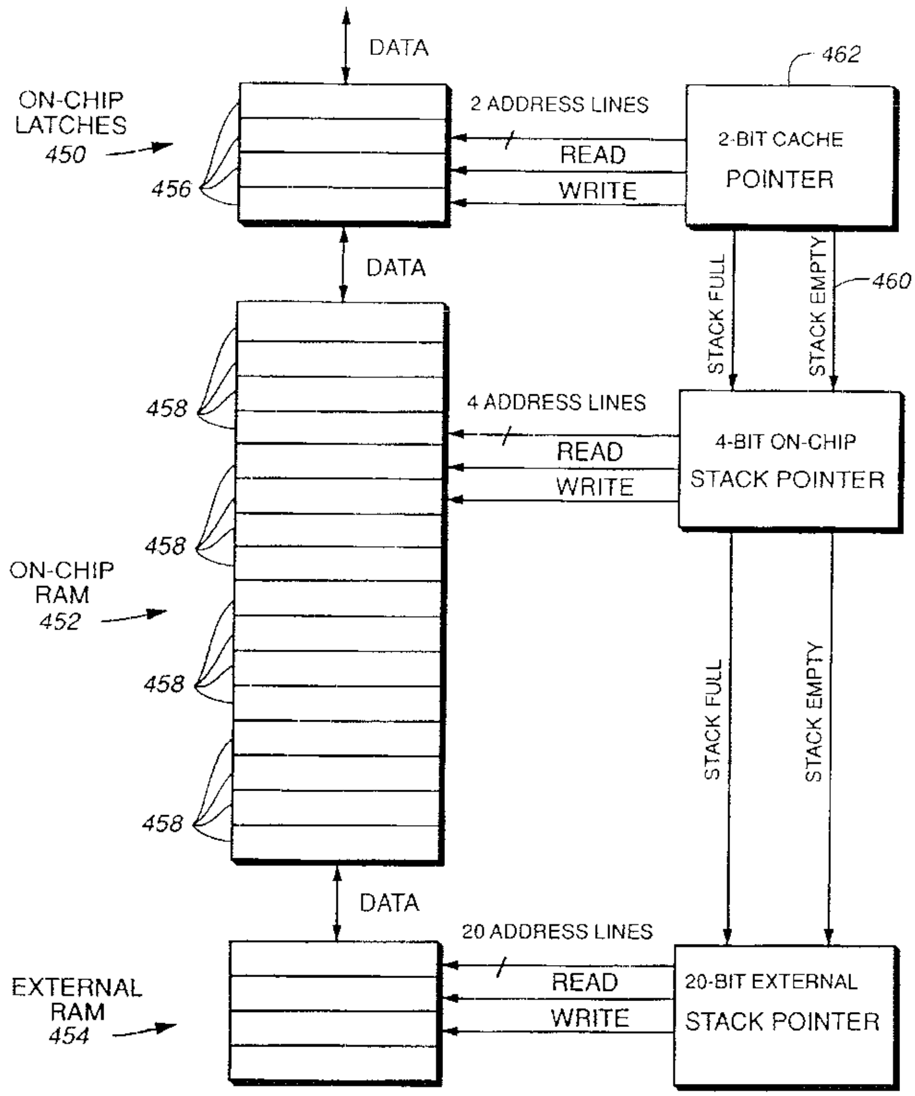
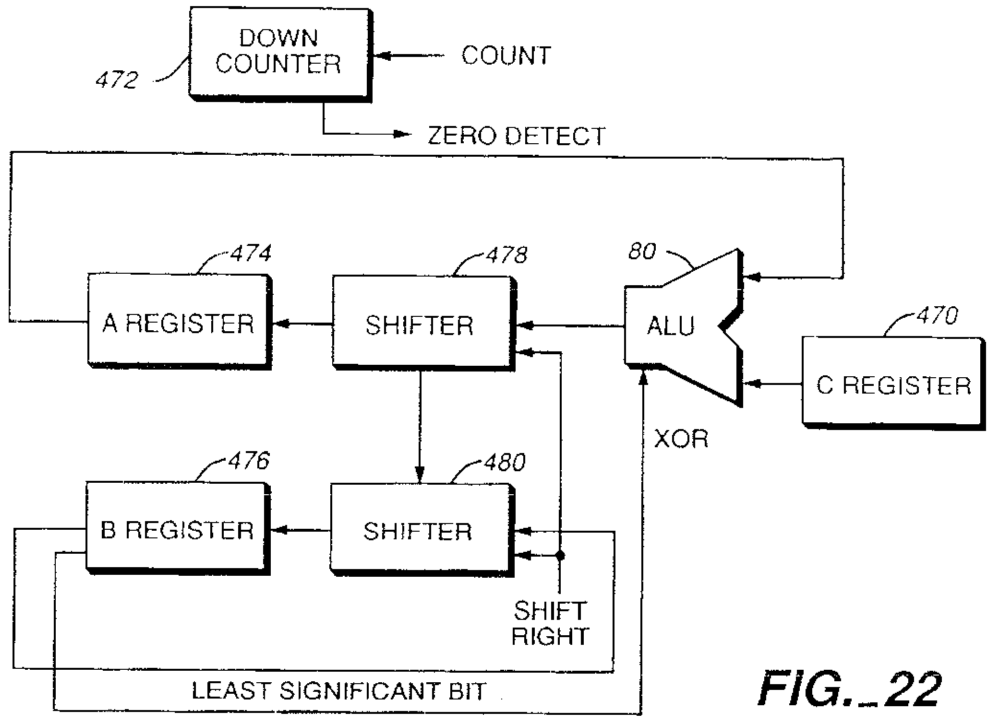
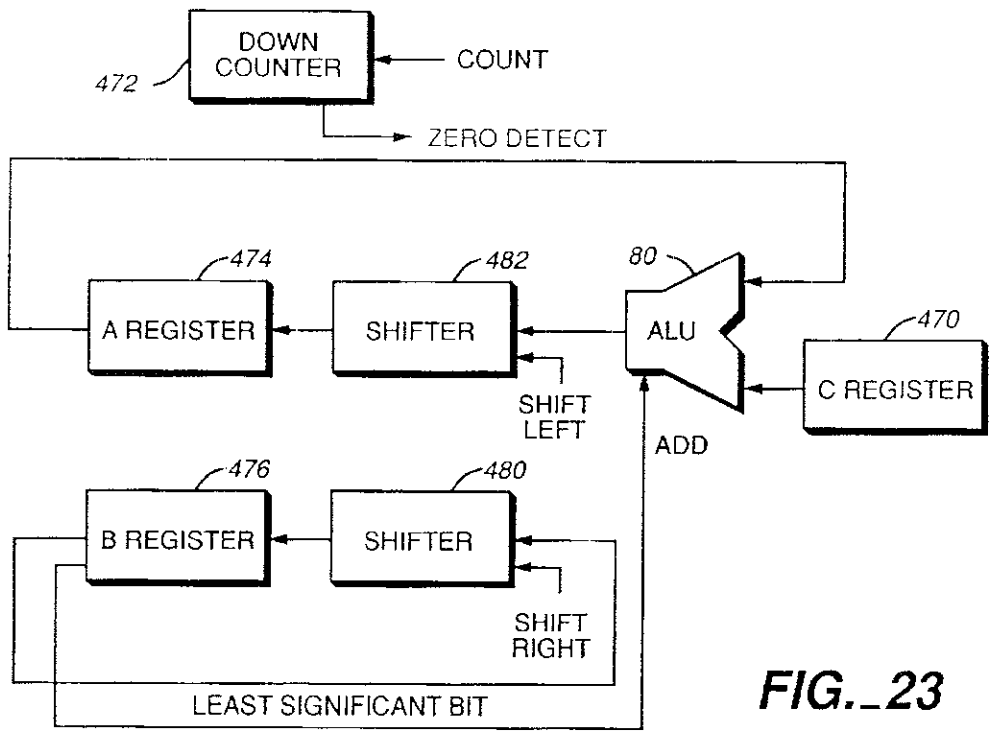

DATA
462
ON-CHIP
LATCHES
450
2 ADDRESS LINES
2-BIT CACHE
POINTER
READ
WRITE
456
DATA
STACK FULL
STACK EMPTY
460
458
4 ADDRESS LINES
4-BIT ON-CHIP
STACK POINTER
READ
WRITE
458
ON-CHIP
RAM
452
458
STACK FULL
STACK EMPTY
458
DATA
20 ADDRESS LINES
EXTERNAL
RAM
454
20-BIT EXTERNAL
STACK POINTER
READ
WRITE

FIG._21



DOWN COUNTER
472
COUNT
ZERO DETECT
474
A REGISTER
478
SHIFTER
80
ALU
470
C REGISTER
XOR
476
B REGISTER
480
SHIFTER
SHIFT RIGHT
LEAST SIGNIFICANT BIT

FIG._22



DOWN COUNTER
472
COUNT
ZERO DETECT
474
A REGISTER
482
SHIFTER
80
ALU
470
C REGISTER
SHIFT LEFT
ADD
476
B REGISTER
480
SHIFTER
SHIFT RIGHT
LEAST SIGNIFICANT BIT

FIG._23

HIGH PERFORMANCE MICROPROCESSOR HAVING VARIABLE SPEED SYSTEM CLOCK

This application is a divisional of U.S. patent application No. 08/484,918, filed Jun. 7, 1995, now U.S. Pat. No. 5,809,336 which is a divisional of U.S. patent application No. 07/389,334, filed Aug. 3, 1989 now U.S. Pat. No. 5,982,231

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to a simplified reduced instruction set computer (RISC) microprocessor. More particularly, it relates to such a microprocessor which is capable of performance levels of, for example, 20 million instructions per second (MIPS) at a price of, for example, 20 dollars.

2. Description of the Prior Art

Since the invention of the microprocessor, improvements in its design have taken two different approaches. In the first approach, a brute force gain in performance has been achieved through the provision of greater numbers of faster transistors in the microprocessor integrated circuit and an instruction set of increased complexity. This approach is exemplified by the Motorola 68000 and Intel 80X86 microprocessor families. The trend in this approach is to larger die sizes and packages, with hundreds of pinouts.

More recently, it has been perceived that performance gains can be achieved through comparative simplicity, both in the microprocessor integrated circuit itself and in its instruction set. This second approach provides RISC microprocessors, and is exemplified by the Sun SPARC and the Intel 8960 microprocessors. However, even with this approach as conventionally practiced, the packages for the microprocessor are large, in order to accommodate the large number of pinouts that continue to be employed. A need therefore remains for further simplification of high performance microprocessors.

With conventional high performance microprocessors, fast static memories are required for direct connection to the microprocessors in order to allow memory accesses that are fast enough to keep up with the microprocessors. Slower dynamic random access memories (DRAMs) are used with such microprocessors only in a hierarchical memory arrangement, with the static memories acting as a buffer between the microprocessors and the DRAMs. The necessity to use static memories increases cost of the resulting systems.

Conventional microprocessors provide direct memory accesses (DMA) for system peripheral units through DMA controllers, which may be located on the microprocessor integrated circuit, or provided separately. Such DMA controllers can provide routine handling of DMA requests and responses, but some processing by the main central processing unit (CPU) of the microprocessor is required.

SUMMARY OF THE INVENTION

Accordingly, it is an object of this invention to provide a microprocessor with a reduced pin count and cost compared to conventional microprocessors.

It is another object of the invention to provide a high performance microprocessor that can be directly connected to DRAMs without sacrificing microprocessor speed.

It is a further object of the invention to provide a high performance microprocessor in which DMA does not require use of the main CPU during DMA requests and responses and which provides very rapid DMA response with predictable response times.

The attainment of these and related objects may be achieved through use of the novel high performance, low cost microprocessor herein disclosed. The microprocessor integrated circuit includes a processing unit disposed upon an integrated circuit substrate. In a preferred implementation the processing unit operates in accordance with a predefined sequence of program instructions stored within an instruction register. A memory capable of storing information provided by the processing unit and occupying a larger area of the integrated circuit substrate than the processing unit is also provided within the microprocessor integrated circuit. The memory may be implemented using, for example, dynamic or static random-access memory.

The attainment of the foregoing and related objects, advantages and features of the invention should be more readily apparent to those skilled in the art, after review of the following more detailed description of the invention, taken together with the drawings, in which:

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is an external plan view of an integrated circuit package incorporating a microprocessor in accordance with the invention.

FIG. 2 is a block diagram of a microprocessor in accordance with the invention.

FIG. 3 is a block diagram of a portion of a data processing system incorporating the microprocessor of FIGS. 1 and 2.

FIG. 4 is a more detailed block diagram of a portion of the microprocessor shown in FIG. 2.

FIG. 5 is a more detailed block diagram of another portion of the microprocessor shown in FIG. 2.

FIG. 6 is a block diagram of another portion of the data processing system shown in part in FIG. 3 and incorporating the microprocessor of FIGS. 1–2 and 4–5.

FIGS. 7 and 8 are layout diagrams for the data processing system shown in part in FIGS. 3 and 6.

FIG. 9 is a layout diagram of a second embodiment of a microprocessor in accordance with the invention in a data processing system on a single integrated circuit.

FIG. 10 is a more detailed block diagram of a portion of the data processing system of FIGS. 7 and 8.

FIG. 11 is a timing diagram useful for understanding operation of the system portion shown in FIG. 12.

FIG. 12 is another more detailed block diagram of a further portion of the data processing system of FIGS. 7 and 8.

FIG. 13 is a more detailed block diagram of a portion of the microprocessor shown in FIG. 2.

FIG. 14 is a more detailed block and schematic diagram of a portion of the system shown in FIGS. 3 and 7–8.

FIG. 15 is a graph useful for understanding operation of the system portion shown in FIG. 14.

FIG. 16 is a more detailed block diagram showing part of the system portion shown in FIG. 4.

FIG. 17 is a more detailed block diagram of a portion of the microprocessor shown in FIG. 2.

FIG. 18 is a more detailed block diagram of part of the microprocessor portion shown in FIG. 17.

FIG. 19 is a set of waveform diagrams useful for understanding operation of the part of the microprocessor portion shown in FIG. 18.

FIG. 20 is a more detailed block diagram showing another part of the system portion shown in FIG. 4.

FIG. 21 is a more detailed block diagram showing another part of the system portion shown in FIG. 4.

FIGS. 22 and 23 are more detailed block diagrams showing another part of the system portion shown in FIG. 4.

DETAILED DESCRIPTION OF THE INVENTION

OVERVIEW

The microprocessor of this invention is desirably implemented as a 32-bit microprocessor optimized for:

HIGH EXECUTION SPEED, and

LOW SYSTEM COST.

In this embodiment, the microprocessor can be thought of as 20 MIPS for 20 dollars. Important distinguishing features of the microprocessor are:

Uses low-cost commodity DYNAMIC RAMS to run 20 MIPS

4 instruction fetch per memory cycle

On-chip fast page-mode memory management

Runs fast without external cache

Requires few interfacing chips

Crams 32-bit CPU in 44 pin SOJ package

The instruction set is organized so that most operations can be specified with 8-bit instructions. Two positive products of this philosophy are:

Programs are smaller,

Programs can execute much faster.

The bottleneck in most computer systems is the memory bus. The bus is used to fetch instructions and fetch and store data. The ability to fetch four instructions in a single memory bus cycle significantly increases the bus availability to handle data.

Turning now to the drawings, more particularly to FIG. 1, there is shown a packaged 32-bit microprocessor **50** in a 44-pin plastic leadless chip carrier, shown approximately 100 times its actual size of about 0.8 inch on a side. The fact that the microprocessor **50** is provided as a 44-pin package represents a substantial departure from typical microprocessor packages, which usually have about 200 input/output (I/O) pins. The microprocessor **50** is rated at 20 million instructions per second (MIPS). Address and data lines **52**, also labelled D0–D31, are shared for addresses and data without speed penalty as a result of the manner in which the microprocessor **50** operates, as will be explained below.

DYNAMIC RAM

In addition to the low cost 44-pin package, another unusual aspect of the high performance microprocessor **50** is that it operates directly with dynamic random access memories (DRAMs), as shown by row address strobe (RAS) and column address strobe (CAS) I/O pins **54**. The other I/O pins for the microprocessor **50** include VDD pins **56**, VSS pins **58**, output enable pin **60**, write pin **62**, clock pin **64** and reset pin **66**.

All high speed computers require high speed and expensive memory to keep up. The highest speed static RAM memories cost as much as ten times as much as slower dynamic RAMs. This microprocessor has been optimized to use low-cost dynamic RAM in high-speed page-mode. Page-mode dynamic RAMs offer static RAM performance without the cost penalty. For example, low-cost 85 nsec. dynamic RAMs access at 25 nsec when operated in fast page-mode. Integrated fast page-mode control on the microprocessor chip simplifies system interfacing and results in a faster system.

Details of the microprocessor **50** are shown in FIG. 2. The microprocessor **50** includes a main central processing unit (CPU) **70** and a separate direct memory access (DMA) CPU **72** in a single integrated circuit making up the microprocessor **50**. The main CPU **70** has a first **16** deep push down stack **74**, which has a top item register **76** and a next item register **78**, respectively connected to provide inputs to an arithmetic logic unit (ALU) **80** by lines **82** and **84**. An output of the ALU **80** is connected to the top item register **76** by line **86**. The output of the top item register at **82** is also connected by line **88** to an internal data bus **90**.

A loop counter **92** is connected to a decrementer **94** by lines **96** and **98**. The loop counter **92** is bidirectionally connected to the internal data bus **90** by line **100**. Stack pointer **102**, return stack pointer **104**, mode register **106** and instruction register **108** are also connected to the internal data bus **90** by lines **110, 112, 114** and **116**, respectively. The internal data bus **90** is connected to memory controller **118** and to gate **120**. The gate **120** provides inputs on lines **122, 124** and **126** to X register **128**, program counter **130** and Y register **132** of return push down stack **134**. The X register **128**, program counter **130** and Y register **132** provide outputs to internal address bus **136** on lines **138, 140** and **142**. The internal address bus provides inputs to the memory controller **118** and to an incrementer **144**. The incrementer **144** provides inputs to the X register, program counter and Y register via lines **146, 122, 124** and **126**. The DMA CPU **72** provides inputs to the memory controller **118** on line **148**. The memory controller **118** is connected to a RAM (not shown) by address/data bus **151** and control lines **153**.

FIG. 2 shows that the microprocessor **50** has a simple architecture. Prior art RISC microprocessors are substantially more complex in design. For example, the SPARC RISC microprocessor has three times the gates of the microprocessor **50**, and the Intel 8960 RISC microprocessor has 20 times the gates of the microprocessor **50**. The speed of this microprocessor is in substantial part due to this simplicity. The architecture incorporates push down stacks and register write to achieve this simplicity.

The microprocessor **50** incorporates an I/O that has been tuned to make heavy use of resources provided on the integrated circuit chip. On chip latches allow use of the same I/O circuits to handle three different things: column addressing, row addressing and data, with a slight to non-existent speed penalty. This triple bus multiplexing results in fewer buffers to expand, fewer interconnection lines, fewer I/O pins and fewer internal buffers.

The provision of on-chip DRAM control gives a performance equal to that obtained with the use of static RAMs. As a result, memory is provided at 1/4 the system cost of static RAM used in most RISC systems.

The microprocessor **50** fetches 4 instructions per memory cycle; the instructions are in an 8-bit format, and this is a 32-bit microprocessor. System speed is therefore 4 times the memory bus bandwidth. This ability enables the microprocessor to break the Von Neumann bottleneck of the speed of getting the next instruction. This mode of operation is possible because of the use of a push down stack and register array. The push down stack allows the use of implied addresses, rather than the prior art technique of explicit addresses for two sources and a destination.

Most instructions execute in 20 nanoseconds in the microprocessor **50**. The microprocessor can therefore execute instructions at 50 peak MIPS without pipeline delays. This is a function of the small number of gates in the microprocessor **50** and the high degree of parallelism in the architecture of the microprocessor.

FIG. 3 shows how column and row addresses are multiplexed on lines D8–D14 of the microprocessor **50** for addressing DRAM **150** from I/O pins **52**. The DRAM **150** is one of eight, but only one DRAM **150** has been shown for clarity. As shown, the lines D11–D18 are respectively connected to row address inputs A0–A8 of the DRAM **150**. Additionally, lines D12–D15 are connected to the data inputs DQ1–DQ4 of the DRAM **150**. The output enable, write and column address strobe pins **54** are respectively connected to the output enable, write and column address strobe inputs of the DRAM **150** by lines **152**. The row address strobe pin **54** is connected through row address strobe decode logic **154** to the row address strobe input of the DRAM **150** by lines **156** and **158**.

D0–D7 pins **52** (FIG. 1) are idle when the microprocessor **50** is outputting multiplexed row and column addresses on D11D18 pins **52**. The D0–D7 pins **52** can therefore simultaneously be used for I/O when right justified I/O is desired. Simultaneous addressing and I/O can therefore be carried out.

FIG. 4 shows how the microprocessor **50** is able to achieve performance equal to the use of static RAMS with DRAMs through multiple instruction fetch in a single clock cycle and instruction fetch-ahead. Instruction register **108** receives four 8-bit byte instruction words 1–4 on 32-bit internal data bus **90**. The four instruction byte **1–4** locations of the instruction register **108** are connected to multiplexer **170** by busses **172**, **174**, **176** and **178**, respectively. A microprogram counter **180** is connected to the multiplexer **170** by lines **182**. The multiplexer **170** is connected to decoder **184** by bus **186**. The decoder **184** provides internal signals to the rest of the microprocessor **50** on lines **188**.

Most significant bits **190** of each instruction byte **1–4** location are connected to a 4-input decoder **192** by lines **194**. The output of decoder **192** is connected to memory controller **118** by line **196**. Program counter **130** is connected to memory controller **118** by internal address bus **136**, and the instruction register **108** is connected to the memory controller **118** by the internal data bus **90**. Address/data bus **198** and control bus **200** are connected to the DRAMS **150** (FIG. 3).

In operation, when the most significant bits **190** of remaining instructions 1–4 are "1" in a clock cycle of the microprocessor **50**, there are no memory reference instructions in the queue. The output of decoder **192** on line **196** requests an instruction fetch ahead by memory controller **118** without interference with other accesses. While the current instructions in instruction register **108** are executing, the memory controller **118** obtains the address of the next set of four instructions from program counter **130** and obtains that set of instructions. By the time the current set of instructions has completed execution, the next set of instructions is ready for loading into the instruction register.

Details of the DMA CPU **72** are provided in FIG. 5. Internal data bus **90** is connected to memory controller **118** and to DMA instruction register **210**. The DMA instruction register **210** is connected to DMA program counter **212** by bus **214**, to transfer size counter **216** by bus **218** and to timed transfer interval counter **220** by bus **222**. The DMA instruction register **210** is also connected to DMA I/O and RAM address register **224** by line **226**. The DMA I/O and RAM address register **224** is connected to the memory controller **118** by memory cycle request line **228** and bus **230**. The DMA program counter **212** is connected to the internal address bus **136** by bus **232**. The transfer size counter **216** is connected to a DMA instruction done decrementer **234** by lines **236** and **238**. The decrementer **234** receives a control input on memory cycle acknowledge line **240**. When transfer size counter **216** has completed its count, it provides a control signal to DMA program counter **212** on line **242**. Timed transfer interval counter **220** is connected to decrementer **244** by lines **246** and **248**. The decrementer **244** receives a control input from a microprocessor system clock on line **250**.

The DMA CPU **72** controls itself and has the ability to fetch and execute instructions. It operates as a co-processor to the main CPU **70** (FIG. 2) for time specific processing.

FIG. 6 shows how the microprocessor **50** is connected to an electrically programmable read only memory (EPROM) **260** by reconfiguring the data lines **52** so that some of the data lines **52** are input lines and some of them are output lines. Data lines **52** D0–D7 provide data to and from corresponding data terminals **262** of the EPROM **260**. Data lines **52** D9–D18 provide addresses to address terminals **264** of the EPROM **260**. Data lines **52** D19–D31 provide inputs from the microprocessor **50** to memory and I/O decode logic **266**. RAS 0/1 control line **268** provides a control signal for determining whether the memory and I/O decode logic provides a DRAM RAS output on line **270** or a column enable output for the EPROM **260** on line **272**. Column address strobe terminal **60** of the microprocessor **50** provides an output enable signal on line **274** to the corresponding terminal **276** of the EPROM **260**.

FIGS. 7 and 8 show the front and back of a one card data processing system **280** incorporating the microprocessor **50**, MSM514258-10 type DRAMs **150** totalling 2 megabytes, a Motorola 50 MegaHertz crystal oscillator clock **282**, I/O circuits **284** and a 27256 type EPROM **260**. The I/O circuits **284** include a 74HC04 type high speed hex inverter circuit **286**, an IDT39C828 type 10-bit inverting buffer circuit **288**, an IDT39C822 type 10-bit inverting register circuit **290** and two IDT39C823 type 9-bit non-inverting register circuits **292**. The card **280** is completed with a MAX12V type DC—DC converter circuit **294**, 34-pin dual AMP type headers **296**, a coaxial female power connector **298**, and a 3-pin AMP right angle header **300**. The card **280** is a low cost, imbeddable product that can be incorporated in larger systems or used as an internal development tool.

The microprocessor **50** is a very high performance (50 MHz) RISC influenced 32-bit CPU designed to work closely with dynamic RAM. Clock for clock, the microprocessor **50** approaches the theoretical performance limits possible with a single CPU configuration. Eventually the microprocessor **50** and any other processor is limited by the bus bandwidth and the number of bus paths. The critical conduit is between the CPU and memory.

One solution to the bus bandwidth/bus path problem is to integrate a CPU directly onto the memory chips, giving every memory a direct bus to the CPU. FIG. 9 shows another microprocessor **310** that is provided integrally with 1 megabit of DRAM **311** in a single integrated circuit **312**. Until the present invention, this solution has not been practical, because most high performance CPUs require from 500,000 to 1,000,000 transistors and enormous die sizes just by themselves. The microprocessor **310** is equivalent to the microprocessor **50** in FIGS. 1–8. The microprocessors **50** and **310** are the most transistor efficient high performance CPUs in existence, requiring fewer than 50,000 transistors for dual processors **70** and **72** (FIG. 2) or **314** and **316** (less memory). The very high speed of the microprocessors **50** and **310** is to a certain extent a function of the small number of active devices. In essence, the less silicon gets in the way, the faster the electrons can get where they are going.

The microprocessor **310** is therefore the only CPU suitable for integration on the memory chip die **312**. Some

simple modifications to the basic microprocessor **50** to take advantage of the proximity to the DRAM array **311** can also increase the microprocessor **50** clock speed by 50 percent and probably more.

The microprocessor **310** core on board the DRAM die **312** provides most of the speed and functionality required for a large group of applications from automotive to peripheral control. However, the integrated CPU **310**/DRAM **311** concept has the potential to redefine significantly the way multiprocessor solutions can solve a spectrum of very compute intensive problems. The CPU **310**/DRAM**311** combination eliminates the Von Neumann bottleneck by distributing it across numerous CPU/DRAM chips **312**. The microprocessor **310** is a particularly good core for multiprocessing, since it was designed with the SDI targeting array in mind, and provisions were made for efficient interprocessor communications.

Traditional multiprocessor implementations have been very expensive in addition to being unable to exploit fully the available CPU horsepower. Multiprocessor systems have typically been built up from numerous board level or box level computers. The result is usually an immense amount of hardware with corresponding, wiring, power consumption and communications problems. By the time the systems are interconnected, as much as 50 percent of the bus speed has been utilized just getting through the interfaces.

In addition, multiprocessor system software has been scarce. A multiprocessor system can easily be crippled by an inadequate load-sharing algorithm in the system software, which allows one CPU to do a great deal of work and the others to be idle. Great strides have been made recently in systems software, and even UNIX V.4 may be enhanced to support multiprocessing. Several commercial products from such manufacturers as DUAL Systems and UNISOFT do a credible job on 68030 type microprocessor systems now.

The microprocessor **310** architecture eliminates most of the interface friction, since up to **64** CPU **310**/RAM **311** processors should be able to intercommunicate without buffers or latches. Each chip **312** has about 40 MIPS raw speed because placing the DRAM **311** next to the CPU **310** allows the microprocessor **310** instruction cycle to be cut in half, compared to the microprocessor **50**. A **64** chip array of these chips **312** is more powerful than any other existing computer. Such an array fits on a 3×5 card, cost less than a FAX machine, and draw about the same power as a small television.

Dramatic changes in price/performance always reshape existing applications and almost always create new ones. The introduction of microprocessors in the mid 1970s created video games, personal computers, automotive computers, electronically controlled appliances, and low cost computer peripherals.

The integrated circuit **312** will find applications in all of the above areas, plus create some new ones. A common generic parallel processing algorithm handles convolution/Fast Fourier Transform (FFT)/pattern recognition. Interesting product possibilities using the integrated circuit **312** include high speed reading machines, real-time speech recognition, spoken language translation, real-time robot vision, a product to identify people by their faces, and an automotive or aviation collision avoidance system.

A real time processor for enhancing high density television (HDTV) images, or compressing the HDTV information into a smaller bandwidth, would be very feasible. The load sharing in HDTV could be very straightforward. Splitting up the task according to color and frame would require 6, 9 or 12 processors. Practical implementation might require 4 meg RAMs integrated with the microprocessor **310**.

The microprocessor **310** has the following specifications:

CONTROL LINES
4—POWER/GROUND
1—CLOCK
32—DATA I/O
4—SYSTEM CONTROL
EXTERNAL MEMORY FETCH
EXTERNAL MEMORY FETCH AUTOINCREMENT X
EXTERNAL MEMORY FETCH AUTOINCREMENT Y
EXTERNAL MEMORY WRITE
EXTERNAL MEMORY WRITE AUTOINCREMENT X
EXTERNAL MEMORY WRITE AUTOINCREMENT Y
EXTERNAL PROM FETCH
LOAD ALL X REGISTERS
LOAD ALL Y REGISTERS
LOAD ALL PC REGISTERS
EXCHANGE X AND Y
INSTRUCTION FETCH
ADD TO PC
ADD TO X
WRITE MAPPING REGISTER
READ MAPPING REGISTER
REGISTER CONFIGURATION
MICROPROCESSOR **310** CPU **316** CORE
COLUMN LATCH1 (1024 BITS) 32×32 MUX
STACK POINTER (16 BITS)
COLUMN LATCH2 (1024 BITS) 32×32 MUX
RSTACK POINTER (16 BITS)
PROGRAM COUNTER 32 BITS
X0 REGISTER 32 BITS (ACTIVATED ONLY FOR ON-CHIP ACCESSES)
Y0 REGISTER 32 BITS (ACTIVATED ONLY FOR ON-CHIP ACCESSES)
LOOP COUNTER 32 BITS
DMA CPU **314** CORE
DMA PROGRAM COUNTER 24 BITS
INSTRUCTION REGISTER 32 BITS
I/O & RAM ADDRESS REGISTER 32 BITS
TRANSFER SIZE COUNTER 12 BITS
INTERVAL COUNTER 12 BITS

To offer memory expansion for the basic chip **312**, an intelligent DRAM can be produced. This chip will be optimized for high speed operation with the integrated circuit **312** by having three on-chip address registers: Program Counter, X Register and Y register. As a result, to access the intelligent DRAM, no address is required, and a total access cycle could be as short as 10 nsec. Each expansion DRAM would maintain its own copy of the three registers and would be identified by a code specifying its memory address. Incrementing and adding to the three registers will actually take place on the memory chips. A maximum of 64 intelligent DRAM peripherals would allow a large system to be created without sacrificing speed by introducing multiplexers or buffers.

There are certain differences between the microprocessor **310** and the microprocessor **50** that arise from providing the microprocessor **310** on the same die **312** with the DRAM **311**. Integrating the DRAM **311** allows architectural changes in the microprocessor **310** logic to take advantage of existing on-chip DRAM **311** circuitry. Row and column design is

inherent in memory architecture. The DRAMs 311 access random bits in a memory array by first selecting a row of 1024 bits, storing them into a column latch and then selecting one of the bits as the data to be read or written.

The time required to access the data is split between the row access and the column access. Selecting data already stored in a column latch is faster than selecting a random bit by at least a factor of six. The microprocessor 310 takes advantage of this high speed by creating a number of column latches and using them as caches and shift registers. Selecting a new row of information may be thought of as performing a 1024-bit read or write with the resulting immense bus bandwidth.

1. The microprocessor 50 treats its 32-bit instruction register 108 (see FIGS. 2 and 4) as a cache for four 8-bit instructions. Since the DRAM 311 maintains a 1024-bit latch for the column bits, the microprocessor 310 treats the column latch as a cache for 128 8-bit instructions. Therefore, the next instruction will almost always be already present in the cache. Long loops within the cache are also possible and more useful than the 4 instruction loops in the microprocessor 50.

2. The microprocessor 50 uses two 16×32-bit deep register arrays 74 and 134 (FIG. 2) for the parameter stack and the return stack. The microprocessor 310 creates two other 1024-bit column latches to provide the equivalent of two 32x32-bit arrays, which can be accessed twice as fast as a register array.

3. The microprocessor 50 has a DMA capability which can be used for I/O to a video shift register. The microprocessor 310 uses yet another 1024-bit column latch as a long video shift register to drive a CRT display directly. For color displays, three on-chip shift registers could also be used. These shift registers can transfer pixels at a maximum of 100 MHz.

4. The microprocessor 50 accesses memory via an external 32-bit bus. Most of the memory 311 for the microprocessor 310 is on the same die 312. External access to more memory is made using an 8-bit bus. The result is a smaller die, smaller package and lower power consumption than the microprocessor 50.

5. The microprocessor 50 consumes about a third of its operating power charging and discharging the I/O pins and associated capacitances. The DRAMs 150 (FIG. 8) connected to the microprocessor 50 dissipate most of their power in the I/O drivers. A microprocessor 310 system will consume about one-tenth the power of a microprocessor 50 system, since having the DRAM 311 next to the processor 310 eliminates most of the external capacitances to be charged and discharged.

6. Multiprocessing means splitting a computing task between numerous processors in order to speed up the solution. The popularity of multiprocessing is limited by the expense of current individual processors as well as the limited interprocessor communications ability. The microprocessor 310 is an excellent multiprocessor candidate, since the chip 312 is a monolithic computer complete with memory, rendering it low-cost and physically compact.

The shift registers implemented with the microprocessor 310 to perform video output can also be configured as interprocessor communication links. The INMOS transputer attempted a similar strategy, but at much lower speed and without the performance benefits inherent in the microprocessor 310 column latch architecture. Serial I/O is a prerequisite for many multiprocessor topologies because of the many neighbor processors which communicate. A cube has 6 neighbors. Each neighbor communicates using these lines:

DATA IN
CLOCK IN
READY FOR DATA
DATA OUT
DATA READY?
CLOCK OUT

A special start up sequence is used to initialize the on-chip DRAM 311 in each of the processors.

The microprocessor 310 column latch architecture allows neighbor processors to deliver information directly to internal registers or even instruction caches of other chips 312. This technique is not used with existing processors, because it only improves performance in a tightly coupled DRAM system.

7. The microprocessor 50 architecture offers two types of looping structures: LOOP-IF-DONE and MICRO-LOOP. The former takes an 8-bit to 24-bit operand to describe the entry point to the loop address. The latter performs a loop entirely within the 4 instruction queue and the loop entry point is implied as the first instruction in the queue. Loops entirely within the queue run without external instruction fetches and execute up to three times as fast as the long loop construct. The microprocessor 310 retains both constructs with a few differences. The microprocessor 310 microloop functions in the same fashion as the microprocessor 50 operation, except the queue is 1024-bits or 128 8-bit instructions long. The microprocessor 310 microloop can therefore contain jumps, branches, calls and immediate operations not possible in the 4 8-bit instruction microprocessor 50 queue.

Microloops in the microprocessor 50 can only perform simple block move and compare functions. The larger microprocessor 310 queue allows entire digital signal processing or floating point algorithms to loop at high speed in the queue.

The microprocessor 50 offers four instructions to redirect execution:

CALL
BRANCH
BRANCH-IF-ZERO
LOOP-IF-NOT-DONE

These instructions take a variable length address operand 8, 16 or 24 bits long. The microprocessor 50 next address logic treats the three operands similarly by adding or subtracting them to the current program counter. For the microprocessor 310, the 16 and 24-bit operands function in the same manner as the 16 and 24-bit operands in the microprocessor 50. The 8-bit class operands are reserved to operate entirely within the instruction queue. Next address decisions can therefore be made quickly, because only 10 bits of addresses are affected, rather than 32. There is no carry or borrow generated past the 10 bits.

8. The microprocessor 310 CPU 316 resides on an already crowded DRAM die 312. To keep chip size as small as possible, the DMA processor 72 of the microprocessor 50 has been replaced with a more traditional DMA controller 314. DMA is used with the microprocessor 310 to perform the following functions:

Video output to a CRT
Multiprocessor serial communications
8-bit parallel I/O

The DMA controller 314 can maintain both serial and parallel transfers simultaneously. The following DMA sources and destinations are supported by the microprocessor 310:

DESCRIPTION	I.O	LINES
1 Video shift register	OUTPUT	1 to 3
2 Multiprocessor serial	BOTH	6 lines/channel
3 8-bit parallel	BOTH	8 data 4 control

The three sources use separate 1024-bit buffers and separate I O pins Therefore all three may be active simultaneously without interference

The microprocessor 310 can be implemented with either a single multiprocessor serial buffer or separate receive and sending buffers for each channel, allowing simultaneous bidirectional communications with six neighbors simultaneously

FIGS 10 and 11 provide details of the PROM DMA used in the microprocessor **50** The microprocessor **50** executes faster than all but the fastest PROMs PROMS are used in a microprocessor **50** system to store program segments and perhaps entire programs The microprocessor **50** provides a feature on power-up to allow programs to be loaded from low-cost, slow speed PROMs into high speed DRAM for execution The logic which performs this function is part of the DMA memory controller **118** The operation is similar to DMA but not identical since four 8-bit bytes must be assembled on the microprocessor **50** chip then written to the DRAM **150**

The microprocessor **50** directly interfaces to DRAM **150** over a triple multiplexed data and address bus **350**, which carries RAS addresses CAS addresses and data The EPROM **260**, on the other hand, is read with non-multiplexed busses The microprocessor **50** therefore has a special mode which unmultiplexes the data and address lines to read 8 bits of EPROM data Four 8-bit bytes are read in this fashion The multiplexed bus **350** is turned back on and the data is written to the DRAM **150**

When the microprocessor **50** detects a RESET condition, the processor stops the main CPU **70** and forces a mode **0** (PROM LOAD) instruction into the DMA CPU **72** instruction register The DMA instruction directs the memory controller to read the EPROM **260** data at 8 times the normal access time for memory Assuming a 50 MHz microprocessor **50** this means an access time of 320 nsec The instruction also indicates:

The selection address of the EPROM **260** to be loaded.

The number of 32-bit words to transfer,

The DRAM **150** address to transfer into

The sequence of activities to transfer one 32-bit word from EPROM **260** to DRAM **150** are:

1 RAS goes low at **352**, latching the EPROM **260** select information from the high order address bits The EPROM **260** is selected

2 Twelve address bits (consisting of what is normally DRAM CAS addresses plus two byte select bits are placed on the bus **350** going to the EPROM **260** address pins These signals will remain on the lines until the data from the EPROM **260** has been read into the microprocessor **50** For the first byte the byte select bits will be binary 00.

3 CAS goes low at **354** enabling the EPROM **260** data onto the lower 8 bits of the external address/data bus **350**. NOTE: It is important to recognize that, during this part of the cycle the lower 8 bits of the external data/address bus are functioning as inputs but the rest of the bus is still acting as outputs.

4 The microprocessor **50** latches these eight least significant bits internally and shifts them 8 bits left to shift them to the next significant byte position

5 Steps 2 3 and 4 are repeated with byte address **01**

6 Steps 2 3 and 4 are repeated with byte address **10**

7 Steps 2 3 and 4 are repeated with byte address **11**

8 CAS goes high at **356** taking the EPROM **260** off the data bus

9. RAS goes high at **358**, indicating the end of the EPROM **260** access

10 RAS goes low at **360** latching the DRAM select information from the high order address bits At the same time the RAS address bits are latched into the DRAM **150** The DRAM **150** is selected

11 CAS goes low at **362** latching the DRAM **150** CAS addresses

12 The microprocessor **50** places the previously latched EPROM **260** 32-bit data onto the external address/data bus **350** W goes low at **364** writing the 32 bits into the DRAM **150**

13 W goes high at **366** CAS goes high at **368** The process continues with the next word

FIG. 12 shows details of the microprocessor **50** memory controller **118** In operation bus requests stay present until they are serviced CPU **70** requests are prioritized at **370** in the order of: 1 Parameter Stack; 2 Return Stack; 3 Data Fetch; 4, Instruction Fetch The resulting CPU request signal and a DMA request signal are supplied as bus requests to bus control **372** which provides a bus grant signal at **374** Internal address bus **136** and a DMA counter **376** provide inputs to a multiplexer **378** Either a row address or a column address are provided as an output to multiplexed address bus **380** as an output from the multiplexer **378**. The multiplexed address bus **380** and the internal data bus **90** provide address and data inputs, respectively, to multiplexer **382** Shift register **384** supplies row address strobe (RAS) 1 and 2 control signals to multiplexer **386** and column address strobe (CAS) 1 and 2 control signals to multiplexer **388** on lines **390** and **392**. The shift register **384** also supplies output enable (OE) and write (W) signals on lines **394** and **396** and a control signal on line **398** to multiplexer **382** The shift register **384** receives a RUN signal on line **400** to generate a memory cycle and supplies a MEMORY READY signal on line **402** when an access is complete

STACK/REGISTER ARCHITECTURE

Most microprocessors use on-chip registers for temporary storage of variables The on-chip registers access data faster than off-chip RAM A few microprocessors use an on-chip push down stack for temporary storage

A stack has the advantage of faster operation compared to on-chip registers by avoiding the necessity to select source and destination registers (A math or logic operation always uses the top two stack items as source and the top of stack as destination) The stack s disadvantage is that it makes some operations clumsy Some compiler activities in particular require on-chip registers for efficiency

As shown in FIG 13, the microprocessor **50** provides both on-chip registers **134** and a stack **74** and reaps the benefits of both

BENEFITS:

1 Stack math and logic is twice as fast as those available on an equivalent register only machine Most programmers and optimizing compilers can take advantage of this feature.

2 Sixteen registers are available for on-chip storage of local variables which can transfer to the stack for computation The accessing of variables is three to four times as fast as available on a strictly stack machine

The combined stack **74**/register **134** architecture has not been used previously due to inadequate understanding by computer designers of optimizing compilers and the mix of transfer versus math/logic instructions.

ADAPTIVE MEMORY CONTROLLER

A microprocessor must be designed to work with small or large memory configurations. As more memory loads are added to the data, address, and control lines, the switching speed of the signals slows down. The microprocessor 50 multiplexes the address/data bus three ways, so timing between the phases is critical. A traditional approach to the problem allocates a wide margin of time between bus phases so that systems will work with small or large numbers of memory chips connected. A speed compromise of as much as 50% is required.

As shown in FIG. 14, the microprocessor 50 uses a feedback technique to allow the processor to adjust memory bus timing to be fast with small loads and slower with large ones. The OUTPUT ENABLE (OE) line 152 from the microprocessor 50 is connected to all memories 150 on the circuit board. The loading on the output enable line 152 to the microprocessor 50 is directly related to the number of memories 150 connected. By monitoring how rapidly OE 152 goes high after a read, the microprocessor 50 is able to determine when the data hold time has been satisfied and place the next address on the bus.

The level of the OE line 152 is monitored by CMOS input buffer 410 which generates an internal READY signal on line 412 to the microprocessor's memory controller. Curves 414 and 416 of the FIG. 15 graph show the difference in rise time likely to be encountered from a lightly to heavily loaded memory system. When the OE line 152 has reached a predetermined level to generate the READY signal, driver 418 generates an OUTPUT ENABLE signal on OE line 152.

SKIP WITHIN THE INSTRUCTION CACHE

The microprocessor 50 fetches four 8-bit instructions each memory cycle and stores them in a 32-bit instruction register 108, as shown in FIG. 16. A class of "test and skip" instructions can very rapidly execute a very fast jump operation within the four instruction cache.

SKIP CONDITIONS:

Always

ACC non-zero

ACC negative

Carry flag equal logic one

Never

ACC equal zero

ACC positive

Carry flag equal logic zero

The SKIP instruction can be located in any of the four byte positions 420 in the 32-bit instruction register 108. If the test is successful, SKIP will jump over the remaining one, two, or three 8-bit instructions in the instruction register 108 and cause the next four-instruction group to be loaded into the register 108.

As shown, the SKIP operation is implemented by resetting the 2-bit microinstruction counter 180 to zero on line 422 and simultaneously latching the next instruction group into the register 108. Any instructions following the SKIP in the instruction register are overwritten by the new instructions and not executed.

The advantage of SKIP is that optimizing compilers and smart programmers can often use it in place of the longer conditional JUMP instruction. SKIP also makes possible microloops which exit when the loop counts down or when the SKIP jumps to the next instruction group. The result is very fast code.

Other machines (such as the PDP-8 and Data General NOVA) provide the ability to skip a single instruction. The microprocessor 50 provides the ability to skip up to three instructions.

MICROLOOP IN THE INSTRUCTION CACHE

The microprocessor 50 provides the MICROLOOP instruction to execute repetitively from one to three instructions residing in the instruction register 108. The microloop instruction works in conjunction with the LOOP COUNTER 92 (FIG. 2) connected to the internal data bus 90. To execute a microloop, the program stores a count in LOOP COUNTER 92. MICROLOOP may be placed in the first, second, third, or last byte 420 of the instruction register 108. If placed in the first position, execution will just create a delay equal to the number stored in LOOP COUNTER 92 times the machine cycle. If placed in the second, third, or last byte 420, when the microloop instruction is executed, it will test the LOOP COUNT for zero. If zero, execution will continue with the next instruction. If not zero, the LOOP COUNTER 92 is decremented and the 2-bit microinstruction counter is cleared, causing the preceding instructions in the instruction register to be executed again.

Microloop is useful for block move and search operations. By executing a block move completely out of the instruction register 108, the speed of the move is doubled, since all memory cycles are used by the move rather than being shared with instruction fetching. Such a hardware implementation of microloops is much faster than conventional software implementation of a comparable function.

OPTIMAL CPU CLOCK SCHEME

The designer of a high speed microprocessor must produce a product which operate over wide temperature ranges, wide voltage swings, and wide variations in semiconductor processing. Temperature, voltage, and process all affect transistor propagation delays. Traditional CPU designs are done so that with the worse case of the three parameters, the circuit will function at the rated clock speed. The result are designs that must be clocked a factor of two slower than their maximum theoretical performance, so they will operate properly in worse case conditions.

The microprocessor 50 uses the technique shown in FIGS. 17–19 to generate the system clock and its required phases. Clock circuit 430 is the familiar "ring oscillator" used to test process performance. The clock is fabricated on the same silicon chip as the rest of the microprocessor 50.

The ring oscillator frequency is determined by the parameters of temperature, voltage, and process. At room temperature, the frequency will be in the neighborhood of 100 MHZ. At 70 degrees Centigrade, the speed will be 50 MHZ. The ring oscillator 430 is useful as a system clock, with its stages 431 producing phase 0-phase 3 outputs 433 shown in FIG. 19, because its performance tracks the parameters which similarly affect all other transistors on the same silicon die. By deriving system timing from the ring oscillator 430, CPU 70 will always execute at the maximum frequency possible, but never too fast. For example, if the processing of a particular die is not good resulting in slow transistors, the latches and gates on the microprocessor 50 will operate slower than normal. Since the microprocessor 50 ring oscillator clock 430 is made from the same transistors on the same die as the latches and gates, it too will operate slower (oscillating at a lower frequency), providing compensation which allows the rest of the chip's logic to operate properly.

ASYNCHRONOUS/SYNCHRONOUS CPU

Most microprocessors derive all system timing from a single clock. The disadvantage is that different parts of the system can slow all operations. The microprocessor 50 provides a dual-clock scheme as shown in FIG. 17, with the CPU 70 operating asynchronously to I/O interface 432 forming part of memory controller 118 (FIG. 2) and the I/O

interface 432 operating synchronously with the external world of memory and I/O devices. The CPU 70 executes at the fastest speed possible using the adaptive ring counter clock 430. Speed may vary by a factor of four depending upon temperature, voltage, and process. The external world must be synchronized to the microprocessor 50 for operations such as video display updating and disc drive reading and writing. This synchronization is performed by the I/O interface 432, speed of which is controlled by a conventional crystal clock 434. The interface 432 processes requests for memory accesses from the microprocessor 50 and acknowledges the presence of I/O data. The microprocessor 50 fetches up to four instructions in a single memory cycle and can perform much useful work before requiring another memory access. By decoupling the variable speed of the CPU 70 from the fixed speed of the I/O interface 432, optimum performance can be achieved by each. Recoupling between the CPU 70 and the interface 432 is accomplished with handshake signals on lines 436, with data/addresses passing on bus 90, 136.

ASYNCHRONOUS/SYNCHRONOUS CPU IMBEDDED ON A DRAM CHIP

System performance is enhanced even more when the DRAM 311 and CPU 314 (FIG. 9) are located on the same die. The proximity of the transistors means that DRAM 311 and CPU 314 parameters will closely follow each other. At room temperature, not only would the CPU 314 execute at 100 MHZ, but the DRAM 311 would access fast enough to keep up. The synchronization performed by the I/O interface 432 would be for DMA and reading and writing I/O ports. In some systems (such as calculators) no I/O synchronization at all would be required, and the I/O clock would be tied to the ring counter clock.

VARIABLE WIDTH OPERANDS

Many microprocessors provide variable width operands. The microprocessor 50 handles operands of 8, 16, or 24 bits using the same op-code. FIG. 20 shows the 32-bit instruction register 108 and the 2-bit microinstruction register 180 which selects the 8-bit instruction. Two classes of microprocessor 50 instructions can be greater than 8-bits, JUMP class and IMMEDIATE. A JUMP or IMMEDIATE op-code is 8-bits, but the operand can be 8, 16, or 24 bits long. This magic is possible because operands must be right justified in the instruction register. This means that the least significant bit of the operand is always located in the least significant bit of the instruction register. The microinstruction counter 180 selects which 8-bit instruction to execute. If a JUMP or IMMEDIATE instruction is decoded, the state of the 2-bit microinstruction counter selects the required 8, 16, or 24 bit operand onto the address or data bus. The unselected 8-bit bytes are loaded with zeros by operation of decoder 440 and gates 442. The advantage of this technique is the saving of a number of op-codes required to specify the different operand sizes in other microprocessors.

TRIPLE STACK CACHE

Computer performance is directly related to the system memory bandwidth. The faster the memories, the faster the computer. Fast memories are expensive, so techniques have been developed to move a small amount of high-speed memory around to the memory addresses where it is needed. A large amount of slow memory is constantly updated by the fast memory, giving the appearance of a large fast memory array. A common implementation of the technique is known as a high-speed memory cache. The cache may be thought of as fast acting shock absorber smoothing out the bumps in memory access. When more memory is required than the shock can absorb, it bottoms out and slow speed memory is accessed. Most memory operations can be handled by the shock absorber itself. The microprocessor 50 architecture has the ALU 80 (FIG. 2) directly coupled to the top two stack locations 76 and 78. The access time of the stack 74 therefore directly affects the execution speed of the processor. The microprocessor 50 stack architecture is particularly suitable to a triple cache technique, shown in FIG. 21 which offers the appearance of a large stack memory operating at the speed of on-chip latches 450. Latches 450 are the fastest form of memory device built on the chip, delivering data in as little as 3 nsec. However latches 450 require large numbers of transistors to construct. On-chip RAM 452 requires fewer transistors than latches, but is slower by a factor of five (15 nsec access). Off-chip RAM 150 is the slowest storage of all. The microprocessor 50 organizes the stack memory hierarchy as three interconnected stacks 450, 452 and 454. The latch stack 450 is the fastest and most frequently used. The on-chip RAM stack 452 is next. The off-chip RAM stack 454 is slowest. The stack modulation determines the effective access time of the stack. If a group of stack operations never push or pull more than four consecutive items on the stack, operations will be entirely performed in the 3 nsec latch stack. When the four latches 456 are filled, the data in the bottom of the latch stack 450 is written to the top of the on-chip RAM stack 452. When the sixteen locations 458 in the on-chip RAM stack 452 are filled, the data in the bottom of the on-chip RAM stack 452 is written to the top of the off-chip RAM stack 454. When popping data off a full stack 450, four pops will be performed before stack empty line 460 from the latch stack pointer 462 transfers data from the on-chip RAM stack 452. By waiting for the latch stack 450 to empty before performing the slower on-chip RAM access, the high effective speed of the latches 456 are made available to the processor. The same approach is employed with the on-chip RAM stack 452 and the off-chip RAM stack 454.

POLYNOMIAL GENERATION INSTRUCTION

Polynomials are useful for error correction, encryption, data compression, and fractal generation. A polynomial is generated by a sequence of shift and exclusive OR operations. Special chips are provided for this purpose in the prior art.

The microprocessor 50 is able to generate polynomials at high speed without external hardware by slightly modifying how the ALU 80 works. As shown in FIG. 22, a polynomial is generated by loading the "order" (also known as the feedback terms) into C Register 470. The value thirty one (resulting in 32 iterations) is loaded into DOWN COUNTER 472. A register 474 is loaded with zero. B register 476 is loaded with the starting polynomial value. When the POLY instruction executes, C register 470 is exclusively ORed with A register 474 if the least significant bit of B register 476 is a one. Otherwise, the contents of the A register 474 passes through the ALU 80 unaltered. The combination of A and B is then shifted right (divided by 2) with shifters 478 and 480. The operation automatically repeats the specified number of iterations, and the resulting polynomial is left in A register 474.

FAST MULTIPLY

Most microprocessors offer a 16×16 or 32×32 bit multiply instruction. Multiply when performed sequentially takes one shift/add per bit, or 32 cycles for 32 bit data. The microprocessor 50 provides a high speed multiply which allows multiplication by small numbers using only a small number of cycles. FIG. 23 shows the logic used to implement the high speed algorithm. To perform a multiply, the size of the multiplier less one is placed in the DOWN COUNTER 472.

For a four bit multiplier, the number three would be stored in the DOWN COUNTER **472**. Zero is loaded into the A register **474**. The multiplier is written bit reversed into the B Register **476**. For example, a bit reversed five (binary 0101) would be written into B as 1010. The multiplicand is written into the C register **470**. Executing the FAST MULT instruction will leave the result in the A Register **474** when the count has been completed. The fast multiply instruction is important because many applications scale one number by a much smaller number. The difference in speed between multiplying a 32×32 bit and a 32×4 bit is a factor of 8. If the least significant bit of the multiplier is a "ONE", the contents of the A register **474** and the C register **470** are added. If the least significant bit of the multiplier is a "ZERO", the contents of the A register are passed through the ALU **80** unaltered. The output of the ALU **80** is shifted left by shifter **482** in each iteration. The contents of the B register **476** are shifted right by the shifter **480** in each iteration.

INSTRUCTION EXECUTION PHILOSOPHY

The microprocessor **50** uses high speed D latches in most of the speed critical areas. Slower on-chip RAM is used as secondary storage.

The microprocessor **50** philosophy of instruction execution is to create a hierarchy of speed as follows:

Logic and D latch transfers	1 cycle	20 nsec
Math	2 cycles	40 nsec
Fetch/store on-chip RAM	2 cycles	40 nsec
Fetch/store in current RAS page	4 cycles	80 nsec
Fetch/store with RAS cycle	11 cycles	220 nsec

With a 50 MHZ clock, many operations can be performed in 20 nsec. and almost everything else in 40 nsec.

To maximize speed, certain techniques in processor design have been used. They include:

Eliminating arithmetic operations on addresses,

Fetching up to four instructions per memory cycle,

Pipelineless instruction decoding

Generating results before they are needed

Use of three level stack caching

PIPELINE PHILOSOPHY

Computer instructions are usually broken down into sequential pieces, for example: fetch, decode, register read, execute, and store. Each piece will require a single machine cycle. In most Reduced Instruction Set Computer (RISC) chips, instruction require from three to six cycles.

RISC instructions are very parallel. For example, each of 70 different instructions in the SPARC (SUN Computer's RISC chip) has five cycles. Using a technique called "pipelining", the different phases of consecutive instructions can be overlapped.

To understand pipelining, think of building five residential homes. Each home will require in sequence, a foundation, framing, plumbing and wiring, roofing, and interior finish. Assume that each activity takes one week. To build one house will take five weeks.

But what if you want to build an entire subdivision? You have only one of each work crew, but when the foundation men finish on the first house, you immediately start them on the second one, and so on. At the end of five weeks, the first home is complete, but you also have five foundations. If you have kept the framing, plumbing, roofing, and interior guys all busy from five weeks on, a new house will be completed each week.

This is the way a RISC chip like SPARC appears to execute an instruction in a single machine cycle. In reality, a RISC chip is executing one fifth of five instructions each machine cycle. And if five instructions stay in sequence, an instruction will be completed each machine cycle.

The problems with a pipeline are keeping the pipe full with instructions. Each time an out of sequence instruction such as a BRANCH or CALL occurs, the pipe must be refilled with the next sequence. The resulting dead time to refill the pipeline can become substantial when many IF/THEN/ELSE statements or subroutines are encountered.

THE PIPELINE APPROACH

The microprocessor **50** has no pipeline as such. The approach of this microprocessor to speed is to overlap instruction fetching with execution of the previously fetched instruction(s). Beyond that, over half the instructions (the most common ones) execute entirely in a single machine cycle of 20 nsec. This is possible because:

1. Instruction decoding resolves in 2.5 nsec.

2. Incremented/decremented and some math values are calculated before they are needed, requiring only a latching signal to execute.

3. Slower memory is hidden from high speed operations by high-speed D latches which access in 4 nsec. The disadvantage for this microprocessor is a more complex chip design process. The advantage for the chip user is faster ultimate throughput since pipeline stalls cannot exist. Pipeline synchronization with availability flag bits and other such pipeline handling is not required by this microprocessor.

For example, in some RISC machines an instruction which tests a status flag may have to wait for up to four cycles for the flag set by the previous instruction to be available to be tested. Hardware and software debugging is also somewhat easier because the user doesn't have to visualize five instructions simultaneously in the pipe.

OVERLAPPING INSTRUCTION FETCH/EXECUTE

The slowest procedure the microprocessor **50** performs is to access memory. Memory is accessed when data is read or written. Memory is also read when instructions are fetched. The microprocessor **50** is able to hide fetch of the next instruction behind the execution of the previously fetched instruction(s). The microprocessor **50** fetches instructions in 4-byte instruction groups. An instruction group may contain from one to four instructions. The amount of time required to execute the instruction group ranges from 4 cycles for simple instructions to 64 cycles for a multiply.

When a new instruction group is fetched, the microprocessor instruction decoder looks at the most significant bit of all four of the bytes. The most significant bit of an instruction determines if a memory access is required. For example, CALL, FETCH, and STORE all require a memory access to execute. If all four bytes have nonzero most significant bits, the microprocessor initiates the memory fetch of the next sequential 4-byte instruction group. When the last instruction in the group finishes executing, the next 4-byte instruction group is ready and waiting on the data bus needing only to be latched into the instruction register. If the 4-byte instruction group required four or more cycles to execute and the next sequential access was a column address strobe (CAS) cycle, the instruction fetch was completely overlapped with execution.

INTERNAL ARCHITECTURE

The microprocessor 50 architecture consists of the following:

PARAMETER STACK <--> ALU <-->	Y REGISTER RETURN STACK
<---32 BITS---> 16 DEEP	<---32 BITS---> 16 DEEP
Used for math and logic.*	Used for subroutine and interrupt return addresses as well as local variables
Push down stack	Push down stack
Can overflow into off-chip RAM	Can overflow into off-chip RAM
	Can also be accessed relative to top of stack
LOOP COUNTER	(32-bits, can decrement by 1)
	Used by class of test and loop instructions
X REGISTER	(32-bits, can increment or decrement by 4)
	Used to point to RAM locations
PROGRAM COUNTER	(32-bits, increments by 4). Points to 4-byte instruction groups in RAM
INSTRUCTION REG	(32-Bits). Holds 4-byte instruction groups while they are being decoded and executed

*Math and logic operations use the TOP item and NEXT to top Parameter Stack items as the operands. The result is pushed onto the Parameter Stack.

**Return addresses from subroutines are placed on the Return Stack. The Y REGISTER is used as a pointer to RAM locations. Since the Y REGISTER is the top item of the Return Stack, nesting of indices is straightforward.

MODE—A register with mode and status bits

MODE-BITS

—Slow down memory accesses by 8 if "1". Run full speed if "0". (Provided for access to slow EPROM.)

—Divide the system clock by 1023 if "1" to reduce power consumption. Run full speed if "0". (On-chip counters slow down if this bit is set.)

—Enable external interrupt 1

—Enable external interrupt 2

—Enable external interrupt 3

—Enable external interrupt 4

—Enable external interrupt 5

—Enable external interrupt 6

—Enable external interrupt 7

ON-CHIP MEMORY LOCATIONS

MODE-BITS

DMA-POINTER

DMA-COUNTER

STACK-POINTER—Pointer into Parameter Stack

STACK-DEPTH—Depth of on-chip Parameter Stack

RSTACK-POINTER—Pointer into Return Stack

RSTACK-DEPTH—Depth of on-chip Return Stack

ADDRESSING MODE HIGH POINTS

The data bus is 32-bits wide. All memory fetches and stores are 32-bits. Memory bus addresses are 30 bits. The least significant 2 bits are used to select one-of-four bytes in some addressing modes. The Program Counter, X Register, and Y Register are implemented as D latches with their outputs going to the memory address bus and the bus incrementer/decrementer. Incrementing one of these registers can happen quickly, because the incremented value has already rippled through the inc/dec logic and need only be clocked into the latch. Branches and Calls are made to 32-bit word boundaries.

INSTRUCTION SET

32-BIT INSTRUCTION FORMAT

The thirty two bit instructions are CALL, BRANCH, BRANCH-IF-ZERO, and LOOP-IF-NOT-DONE. These instructions require the calculation of an effective address. In many computers, the effective address is calculated by adding or subtracting an operand with the current Program Counter. This math operation requires from four to seven machine cycles to perform and can definitely bog down machine execution. The microprocessor's strategy is to perform the required math operation at assembly or linking time and do a much simpler "Increment to next page" or "Decrement to previous page" operation at run time. As a result, the microprocessor branches execute in a single cycle.

24-BIT OPERAND FORM

Byte 1	Byte 2	Byte 3	Byte 4
WWWWWW XX -	YYYYYYYY -	YYYYYYYY -	YYYYYYYY

With a 24-bit operand, the current page is considered to be defined by the most significant 6 bits of the Program Counter.

16-BIT OPERAND FORM:

QQQQQQQQ - WWWWWW XX - YYYYYYYY - YYYYYYYY With a 16-bit operand, the current page is considered to be defined by the most significant 14 bits of the Program Counter.

8-BIT OPERAND FORM:

QQQQQQQQ - QQQQQQQQ - WWWWWW XX - YYYYYYYY With an 8-bit operand, the current page is considered to be defined by the most significant 22 bits of the Program Counter.

QQQQQQQQ—Any 8-bit instruction.

WWWWWW—Instruction op-code.

XX—Select how the address bits will be used:

00—Make all high-order bits zero. (Page zero addressing)

01—Increment the high-order bits. (Use next page)

10—Decrement the high-order bits. (Use previous page)

11—Leave the high-order bits unchanged. (Use current page)

YYYYYYYY—The address operand field. This field is always shifted left two bits (to generate a word rather than byte address) and loaded into the Program Counter. The microprocessor instruction decoder figures out the width of the operand field by the location of the instruction op-code in the four bytes.

The compiler or assembler will normally use the shortest operand required to reach the desired address so that the leading bytes can be used to hold other instructions. The effective address is calculated by combining:

The current Program Counter,

The 8, 16, or 24 bit address operand in the instruction,

Using one of the four allowed addressing modes.

EXAMPLES OF EFFECTIVE ADDRESS CALCULATION

EXAMPLE 1

Byte 1	Byte 2	Byte 3	Byte 4
QQQQQQQQ	QQQQQQQQ	00000011	10011000

The "QQQQQQQQs" in Byte 1 and 2 indicate space in the 4-byte memory fetch which could be hold two other instructions to be executed prior to the CALL instruction. Byte 3 indicates a CALL instruction (six zeros) in the current page (indicated by the 11 bits). Byte 4 indicates that the hexadecimal number **98** will be forced into the Program Counter bits **2** through **10**. (Remember a CALL or BRANCH always goes to a word boundary so the two least significant bits are always set to zero). The effect of this instruction would be to CALL a subroutine at WORD location HEX **98** in the current page. The most significant 22 bits of the Program Counter define the current page and will be unchanged.

EXAMPLE 2

Byte 1	Byte 2	Byte 3	Byte 4
00000101	00000001	00000000	00000000

If we assume that the Program Counter was HEX 0000 0156 which is binary: 00000000 00000000 00000001 01010110 =OLD PROGRAM COUNTER. Byte **1** indicates a BRANCH instruction op code (000001) and "01" indicates select the next page. Byte **2**,**3**, and **4** are the address operand. These 24-bits will be shifted to the left two places to define a WORD address. HEX 0156 shifted left two places is HEX 0558. Since this is a 24-bit operand instruction, the most significant 6 bits of the Program Counter define the current page. These six bits will be incremented to select the next page. Executing this instruction will cause the Program Counter to be loaded with HEX 0400 0558 which is binary:

00000100 00000000 00000101 01011000 =NEW PROGRAM COUNTER

INSTRUCTIONS

CALL-LONG

0000 00XX - YYYYYYYY - YYYYYYYY - YYYYYYYY

Load the Program Counter with the effective WORD address specified. Push the current PC contents onto the RETURN STACK.

OTHER EFFECTS: CARRY or modes, no effect. May cause Return Stack to force an external memory cycle if on-chip Return Stack is full.

BRANCH

0000 01XX - YYYYYYYY - YYYYYYYY - YYYYYYYY

Load the Program Counter with the effective WORD address specified.

OTHER EFFECTS: NONE

BRANCH-IF-ZERO

0000 10XX - YYYYYYYY - YYYYYYYY - YYYYYYYY

Test the TOP value on the Parameter Stack. If the value is equal to zero, load the Program Counter with the effective WORD address specified. If the TOP value is not equal to zero, increment the Program Counter and fetch and execute the next instruction.

OTHER EFFECTS: NONE

LOOP-IF-NOT-DONE

0000 11 YY - (XXXX XXXX) - (XXXX XXXX) - (XXXX XXXX)

If the LOOP COUNTER is not zero, load the Program Counter with the effective WORD address specified. If the LOOP COUNTER is zero, decrement the LOOP COUNTER, increment the Program Counter and fetch and execute the next instruction.

OTHER EFFECTS: NONE

8-BIT INSTRUCTIONS PHILOSOPHY

Most of the work in the microprocessor **50** is done by the 8-bit instructions. Eight bit instructions are possible with the microprocessor because of the extensive use of implied stack addressing. Many 32-bit architectures use 8-bits to specify the operation to perform but use an additional 24-bits to specify two sources and a destination.

For math and logic operations, the microprocessor **50** exploits the inherent advantage of a stack by designating the source operand(s) as the top stack item and the next stack item. The math or logic operation is performed, the operands are popped from the stack, and the result is pushed back on the stack. The result is a very efficient utilization of instruction bits as well as registers. A comparable situation exists between Hewlett Packard calculators (which use a stack) and Texas Instrument calculators which don't. The identical operation on an HP will require one half to one third the keystrokes of the TI.

The availability of 8-bit instructions also allows another architectural innovation, the fetching of four instructions in a single 32-bit memory cycle. The advantages of fetching multiple instructions are:

Increased execution speed even with slow memories.

Similar performance to the Harvard (separate data and instruction busses) without the expense.

Opportunities to optimize groups of instructions.

The capability to perform loops within this mini-cache.

The microloops inside the four instruction group are effective for searches and block moves.

SKIP INSTRUCTIONS

The microprocessor **50** fetches instructions in 32-bit chunks called 4-byte instruction groups. These four bytes may contain four 8-bit instructions or some mix of 8-bit and 16 or 24-bit instructions. SKIP instructions in the microprocessor skip any remaining instructions in a 4-byte instruction group and cause a memory fetch to get the next 4-byte instruction group. Conditional SKIPs when combined with 3-byte BRANCHES will create conditional BRANCHES. SKIPs may also be used in situations when no use can be made of the remaining bytes in a 4-instruction group. A SKIP executes in a single cycle, whereas a group of three NOPs would take three cycles.

SKIP-ALWAYS—Skip any remaining instructions in this 4-byte instruction group.

Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group.

SKIP-IF-ZERO—If the TOP item of the Parameter Stack is zero, skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group.

If the TOP item is not zero, execute the next sequential instruction.

SKIP-IF-POSITIVE—If the TOP item of the Parameter Stack has a the most significant bit (the sign bit) equal to "0", skip any remaining instructions in the 4-byte instruction

group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the TOP item is not "0", execute the next sequential instruction.

SKIP-IF-NO-CARRY—If the CARRY flag from a SHIFT or arithmetic operation is not equal to "1" skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the CARRY is equal to "1", execute the next sequential instruction.

SKIP-NEVER—Execute the next sequential (NOP) instruction. (Delay one machine cycle).

SKIP-IF-NOT-ZERO - If the TOP item on the Parameter Stack is not equal to "0", skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group.

If the TOP item is equal "0", execute the next sequential instruction.

SKIP-IF-NEGATIVE—If the TOP item on the Parameter Stack has its most significant bit (sign bit) set to "1", skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the TOP item has its most significant bit set to "0", execute the next sequential instruction.

SKIP-IF-CARRY—If the CARRY flag is set to "1" as a result of SHIFT or arithmetic operation, skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the CARRY flag is "0", execute the next sequential instruction.

MICROLOOPS

Microloops are a unique feature of the microprocessor architecture which allows controlled looping within a 4-byte instruction group. A microloop instruction tests the LOOP COUNTER for "0" and may perform an additional test. If the LOOP COUNTER is not "0" and the test is met, instruction execution continues with the first instruction in the 4-byte instruction group, and the LOOP COUNTER is decremented. A microloop instruction will usually be the last byte in a 4-byte instruction group, but it can be any byte. If the LOOP COUNTER is "0" or the test is not met, instruction execution continues with the next instruction. If the microloop is the last byte in the 4-byte instruction group, the most significant 30-bits of the Program Counter are incremented and the next 4-byte instruction group is fetched from memory. On a termination of the loop on LOOP COUNTER equal to "0", the LOOP COUNTER will remain at "0". Microloops allow short iterative work such as moves and searches to be performed without slowing down to fetch instructions from memory.

EXAMPLE

Byte 1 FETCH-VIA-X-AUTOINCREMENT	Byte 2 STORE-VIA-Y-AUTOINCREMENT
Byte 3 ULOOP-UNTIL-DONE	Byte 4 QQQQQQQQ

This example will perform a block move. To initiate the transfer, X will be loaded with the starting address of the source. Y will be loaded with the starting address of the destination. The LOOP COUNTER will be loaded with the number of 32-bit words to move. The microloop will FETCH and STORE and count down the LOOP COUNTER until it reaches zero. QQQQQQQQ indicates any instruction can follow.

MICROLOOP INSTRUCTIONS

ULOOP-UNTIL-DONE—If the LOOP COUNTER is not "0", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0", continue execution with the next instruction.

ULOOP-IF-ZERO—If the LOOP COUNTER is not "0" and the TOP item on the Parameter Stack is "0", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the TOP item is "1", continue execution with the next instruction.

ULOOP-IF-POSITIVE—If the LOOP COUNTER is not "0" and the most significant bit (sign bit) is "0", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the TOP item is "1", continue execution with the next instruction.

ULOOP-IF-NOT-CARRY-CLEAR—If the LOOP COUNTER is not "0" and the floating point exponents found in TOP and NEXT are not aligned, continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the exponents are aligned, continue execution with the next instruction.

This instruction is specifically designed for combination with special SHIFT instructions to align two floating point numbers.

ULOOP-NEVER—(DECREMENT-LOOP-COUNTER) Decrement the LOOP COUNTER. Continue execution with the next instruction.

ULOOP-IF-NOT-ZERO—If the LOOP COUNTER is not "0" and the TOP item of the Parameter Stack is "0", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the TOP item is "1", continue execution with the next instruction.

ULOOP-IF-NEGATIVE—If the LOOP COUNTER is not "0" and the most significant bit (sign bit) of the TOP item of the Parameter Stack is "1", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the most significant bit of the Parameter Stack is "0", continue execution with the next instruction.

ULOOP-IF-CARRY-SET—If the LOOP COUNTER is not "0" and the exponents of the floating point numbers found in TOP and NEXT are not aligned, continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the

LOOP COUNTER is "0" or the exponents are aligned continue execution with the next instruction.

RETURN FROM SUBROUTINE OR INTERRUPT

Subroutine calls and interrupt acknowledgements cause a redirection of normal program execution. In both cases, the current Program Counter is pushed onto the Return Stack, so the microprocessor can return to its place in the program after executing the subroutine or interrupt service routine.

NOTE: When a CALL to subroutine or interrupt is acknowledged the Program Counter has already been incremented and is pointing to the 4-byte instruction group following the 4-byte group currently being executed. The instruction decoding logic allows the microprocessor to perform a test and execute a return conditional on the outcome of the test in a single cycle. A RETURN pops an address from the Return Stack and stores it to the Program Counter.

RETURN INSTRUCTIONS

RETURN-ALWAYS—Pop the top item from the Return Stack and transfer it to the Program Counter.

RETURN-IF-ZERO—If the TOP item on the Parameter Stack is "0", pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

RETURN-IF-POSITIVE—If the most significant bit (sign bit) of the TOP item on the Parameter Stack is a "0", pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

RETURN-IF-CARRY-CLEAR—If the exponents of the floating point numbers found in TOP and NEXT are not aligned, pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

RETURN-NEVER—Execute the next instruction.

(NOP)

RETURN-IF-NOT-ZERO—If the TOP item on the Parameter Stack is not "0", pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

RETURN-IF-NEGATIVE—If the most significant bit (sign bit) of the TOP item on the Parameter Stack is a "1", pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

RETURN-IF-CARRY-SET—If the exponents of the floating point numbers found in TOP and NEXT are aligned, pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

HANDLING MEMORY FROM DYNAMIC RAM

The microprocessor 50, like any RISC type architecture, is optimized to handle as many operations as possible on-chip for maximum speed. External memory operations take from 80 nsec. to 220 nsec. compared with on-chip memory speeds of from 4 nsec. to 30 nsec. There are times when external memory must be accessed.

External memory is accessed using three registers:

X-REGISTER—A 30-bit memory pointer which can be used for memory access and simultaneously incremented or decremented.

Y-REGISTER—A 30-bit memory pointer which can be used for memory access and simultaneously incremented or decremented.

PROGRAM-COUNTER—A 30-bit memory pointer normally used to point to 4-byte instruction groups. External memory may be accessed at addresses relative to the PC. The operands are sometimes called "Immediate" or "Literal" in other computers. When used as memory pointer, the PC is also incremented after each operation.

MEMORY LOAD & STORE INSTRUCTIONS

FETCH-VIA-X—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack. X is unchanged.

FETCH-VIA-Y—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack. Y is unchanged.

FETCH-VIA-X-AUTOINCREMENT—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack. After fetching, increment the most significant 30 bits of X to point to the next 32-bit word address.

FETCH-VIA-Y-AUTOINCREMENT—Fetch the 32-bit memory content pointed to by Y and push it onto the Parameter Stack. After fetching, increment the most significant 30 bits of Y to point to the next 32-bit word address.

FETCH-VIA-X-AUTODECREMENT—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack. After fetching, decrement the most significant 30 bits of X to point to the previous 32-bit word address.

FETCH-VIA-Y-AUTODECREMENT—Fetch the 32-bit memory content pointed to by Y and push it onto the Parameter Stack. After fetching, decrement the most significant 30 bits of Y to point to the previous 32-bit word address.

STORE-VIA-X—Pop the top item of the Parameter Stack and store it in the memory location pointed to by X. X is unchanged.

STORE-VIA-Y—Pop the top item of the Parameter Stack and store it in the memory location pointed to by Y. Y is unchanged.

STORE-VIA-X-AUTOINCREMENT—Pop the top item of the Parameter Stack and store it in the memory location pointed to by X. After storing, increment the most significant 30 bits of X to point to the next 32-bit word address.

STORE-VIA-Y-AUTOINCREMENT—Pop the top item of the Parameter Stack and store it in the memory location pointed to by Y. After storing, increment the most significant 30 bits of Y to point to the next 32-bit word address.

STORE-VIA-X-AUTODECREMENT—Pop the top item of the Parameter Stack and store it in the memory location pointed to by X. After storing, decrement the most significant 30 bits of X to point to the previous 32-bit word address.

STORE-VIA-Y-AUTODECREMENT—Pop the top item of the Parameter Stack and store it in the memory location pointed to by Y. After storing, decrement the most significant 30 bits of Y to point to the previous 32-bit word address.

FETCH-VIA-PC—Fetch the 32-bit memory content pointed to by the Program Counter and push it onto the Parameter Stack. After fetching, increment the most significant 30 bits of the Program Counter to point to the next 32-bit word address.

*NOTE When this instruction executes, the PC is pointing to the memory location following the instruction. The effect

is of loading a 32-bit immediate operand. This is an 8-bit instruction and therefore will be combined with other 8-bit instructions in a 4-byte instruction fetch. It is possible to have from one to four FETCH-VIA-PC instructions in a 4-byte instruction fetch. The PC increments after each execution of FETCH-VIA-PC, so it is possible to push four immediate operands on the stack. The four operands would be the found in the four memory locations following the instruction.

BYTE-FETCH-VIA-X—Fetch the 32-bit memory content pointed to by the most significant 30 bits of X. Using the two least significant bits of X, select one of four bytes from the 32-bit memory fetch, right justify the byte in a 32-bit field and push the selected byte preceded by leading zeros onto the Parameter Stack.

BYTE-STORE-VIA-X—Fetch the 32-bit memory content pointed to by the most significant 30 bits of X. Pop the TOP item from the Parameter Stack. Using the two least significant bits of X place the least significant byte into the 32-bit memory data and write the 32-bit entity back to the location pointed to by the most significant 30 bits of X.

OTHER EFFECTS OF MEMORY ACCESS INSTRUCTIONS:

Any FETCH instruction will push a value on the Parameter Stack **74**. If the on-chip stack is full, the stack will overflow into off-chip memory stack resulting in an additional memory cycle. Any STORE instruction will pop a value from the Parameter Stack **74**. If the on-chip stack is empty, a memory cycle will be generated to fetch a value from off-chip memory stack.

HANDLING ON-CHIP VARIABLES

High-level languages often allow the creation of LOCAL VARIABLES. These variables are used by a particular procedure and discarded. In cases of nested procedures, layers of these variables must be maintained. On-chip storage is up to five times faster than off-chip RAM, so a means of keeping local variables on-chip can make operations run faster. The microprocessor **50** provides the capability for both on-chip storage of local variables and nesting of multiple levels of variables through the Return Stack.

The Return Stack **134** is implemented as 16 on-chip RAM locations. The most common use for the Return Stack **134** is storage of return addresses from subroutines and interrupt calls. The microprocessor allows these 16 locations to also be used as addressable registers. The 16 locations may be read and written by two instructions which indicate a Return Stack relative address from 0–15. When high-level procedures are nested, the current procedure variables push the previous procedure variables further down the Return Stack **134**. Eventually, the Return Stack will automatically overflow into off-chip RAM.

ON-CHIP VARIABLE INSTRUCTIONS

READ-LOCAL-VARIABLE XXXX—Read the XXXXth location relative to the top of the Return Stack. (XXXX is a binary number from 0000–1111). Push the item read onto the Parameter Stack. OTHER EFFECTS: If the Parameter Stack is full, the push operation will cause a memory cycle to be generated as one item of the stack is automatically stored to external RAM. The logic which selects the location performs a modulo **16** subtraction. If four local variables have been pushed onto the Return Stack, and an instruction attempts to READ the fifth item, unknown data will be returned.

WRITE-LOCAL-VARIABLE XXXX—Pop the TOP item of the Parameter Stack and write it into the XXXXth location relative to the top of the Return Stack. (XXXX is a binary number from 0000–1111.) OTHER EFFECTS: If the Parameter Stack is empty, the pop operation will cause a memory cycle to be generated to fetch the Parameter Stack item **30** from external RAM. The logic which selects the location performs a modulo **16** subtraction. If four local variables have been pushed onto the Return Stack, and an instruction attempts to WRITE to the fifth item, it is possible to clobber return addresses or wreak other havoc.

REGISTER AND FLIP-FLOP TRANSFER AND PUSH INSTRUCTIONS

DROP—Pop the TOP item from the Parameter Stack and discard it.

SWAP—Exchange the data in the TOP Parameter Stack location with the data in the NEXT Parameter Stack location.

DUP—Duplicate the TOP item on the Parameter Stack and push it onto the Parameter Stack.

PUSH-LOOP-COUNTER—Push the value in LOOP COUNTER onto the Parameter Stack.

POP-RSTACK-PUSH-TO-STACK—Pop the top item from the Return Stack and push it onto the Parameter Stack.

PUSH-X-REG—Push the value in the X Register onto the Parameter Stack.

PUSH-STACK-POINTER—Push the value of the Parameter Stack pointer onto the Parameter Stack.

PUSH-RSTACK-POINTER—Push the value of the Return Stack pointer onto the Return Stack.

PUSH-MODE-BITS—Push the value of the MODE REGISTER onto the Parameter Stack.

PUSH-INPUT—Read the 10 dedicated input bits and push the value (right justified and padded with leading zeros) onto the Parameter Stack.

SET-LOOP-COUNTER—Pop the TOP value from the Parameter Stack and store it into LOOP COUNTER.

POP-STACK-PUSH-TO-RSTACK—Pop the TOP item from the Parameter Stack and push it onto the Return Stack.

SET-X-REG—Pop the TOP item from the Parameter Stack and store it into the X Register.

SET-STACK-POINTER—Pop the TOP item from the Parameter Stack and store it into the Stack Pointer.

SET-RSTACK-POINTER—Pop the TOP item from the Parameter Stack and store it into the Return Stack Pointer.

SET-MODE-BITS—Pop the TOP value from the Parameter Stack and store it into the MODE BITS.

SET-OUTPUT—Pop the TOP item from the Parameter Stack and output it to the 10 dedicated output bits. OTHER EFFECTS: Instructions which push or pop the Parameter Stack or Return Stack may cause a memory cycle as the stacks overflow back and forth between on-chip and off-chip memory.

LOADING A SHORT LITERAL

A special case of register transfer instruction is used to push an 8-bit literal onto the Parameter Stack. This instruction requires that the 8-bits to be pushed reside in the last byte of a 4-byte instruction group. The instruction op-code loading the literal may reside in ANY of the other three bytes in the instruction group.

EXAMPLE

BYTE 1	BYTE 2	BYTE 3
LOAD-SHORT-LITERAL	QQQQQQQQ	QQQQQQQQ
BYTE 4		
00001111		

In this example QQQQQQQQ indicates any other 8-bit instruction. When Byte 1 is executed, binary 00001111 (HEX 0F) from Byte 4 will be pushed (right justified and padded by leading zeros) onto the Parameter Stack. Then the instructions in Byte 2 and Byte 3 will execute. The microprocessor instruction decoder knows not to execute Byte 4. It is possible to push three identical 8-bit values as follows.

BYTE 1	BYTE 2
LOAD-SHORT-LITERAL	LOAD-SHORT-LITERAL
BYTE 3	BYTE 4
LOAD-SHORT-LITERAL	00001111

SHORT-LITERAL-INSTRUCTION

LOAD-SHORT-LITERAL—Push the 8-bit value found in Byte 4 of the current 4-byte instruction group onto the Parameter Stack.

LOGIC INSTRUCTIONS

Logical and math operations use the stack for the source of one or two operands and as the destination for results. The stack organization is a particularly convenient arrangement for evaluating expressions. TOP indicates the top value on the Parameter Stack 74. NEXT indicates the next to top value on the Parameter Stack 74.

AND—Pop TOP and NEXT from the Parameter Stack, perform the logical AND operation on these two operands, and push the result onto the Parameter Stack.

OR—Pop TOP and NEXT from the Parameter Stack, perform the logical OR operation on these two operands, and push the result onto the Parameter Stack.

XOR—Pop TOP and NEXT from the Parameter Stack, perform the logical exclusive OR on these two operands, and push the result onto the Parameter Stack.

BIT-CLEAR—Pop TOP and NEXT from the Parameter Stack, toggle all bits in NEXT, perform the logical AND operation on TOP, and push the result onto the Parameter Stack. (Another way of understanding this instruction is thinking of it as clearing all bits in TOP that are set in NEXT.)

MATH INSTRUCTIONS

Math instruction pop the TOP item and NEXT to top item of the Parameter Stack 74 to use as the operands. The results are pushed back on the Parameter Stack. The CARRY flag is used to latch the "33rd bit" of the ALU result.

ADD—Pop the TOP item and NEXT to top item from the Parameter Stack, add the values together and push the result back on the Parameter Stack. The CARRY flag may be changed.

ADD-WITH-CARRY—Pop the TOP item and the NEXT to top item from the Parameter Stack, add the values together. If the CARRY flag is "1" increment the result. Push the ultimate result back on the Parameter Stack. The CARRY flag may be changed.

ADD-X—Pop the TOP item from the Parameter Stack and read the third item from the top of the Parameter Stack. Add the values together and push the result back on the Parameter Stack. The CARRY flag may be changed.

SUB—Pop the TOP item and NEXT to top item from the Parameter Stack. Subtract NEXT from TOP and push the result back on the Parameter Stack. The CARRY flag may be changed.

SUB-WITH-CARRY—Pop the TOP item and NEXT to top item from the Parameter Stack. Subtract NEXT from TOP. If the CARRY flag is "1", increment the result. Push the ultimate result back on the Parameter Stack. The CARRY flag may be changed.

SUB-X—

SIGNED-MULT-STEP—

UNSIGNED-MULT-STEP—

SIGNED-FAST-MULT—

FAST-MULT-STEP—

UNSIGNED-DIV-STEP—

GENERATE-POLYNOMIAL—

ROUND—

COMPARE—Pop the TOP item and NEXT to top item from the Parameter Stack. Subtract NEXT from TOP. If the result has the most significant bit equal to "0" (the result is positive), push the result onto the Parameter Stack. If the result has the most significant bit equal to "1" (the result is negative), push the old value of TOP onto the Parameter Stack. The CARRY flag may be affected.

SHIFT/ROTATE

SHIFT-LEFT—Shift the TOP Parameter Stack item left one bit. The CARRY flag is shifted into the least significant bit of TOP.

SHIFT-RIGHT—Shift the TOP Parameter Stack item right one bit. The least significant bit of TOP is shifted into the CARRY flag. Zero is shifted into the most significant bit of TOP.

DOUBLE-SHIFT-LEFT—Treating the TOP item of the Parameter Stack as the most significant word of a 64-bit number and the NEXT stack item as the least significant word, shift the combined 64-bit entity left one bit. The CARRY flag is shifted into the least significant bit of NEXT.

DOUBLE-SHIFT-RIGHT—Treating the TOP item of the Parameter Stack as the most significant word of a 64-bit number and the NEXT stack item as the least significant word, shift the combined 64-bit entity right one bit. The least significant bit of NEXT is shifted into the CARRY flag. Zero is shifted into the most significant bit of TOP.

OTHER INSTRUCTIONS

FLUSH-STACK—Empty all on-chip Parameter Stack locations into off-chip RAM. (This instruction is useful for multitasking applications.) This instruction accesses a counter which holds the depth of the on-chip stack and can require from none to 16 external memory cycles.

FLUSH-RSTACK—Empty all on-chip Return Stack locations into off-chip RAM. (This instruction is useful for multitasking applications.) This instruction accesses a counter which holds the depth of the on-chip Return Stack and can require from none to 16 external memory cycles.

It should further be apparent to those skilled in the art that various changes in form and details of the invention as

shown and described may be made. It is intended that such changes be included within the spirit and scope of the claims appended hereto.

What is claimed is:

1. A microprocessor integrated circuit comprising:

a program-controlled processing unit operative in accordance with a sequence of program instructions;

a memory coupled to said processing unit and capable of storing information provided by said processing unit;

a plurality of column latches coupled to the processing unit and the memory, wherein, during a read operation, a row of bits are read from the memory and stored in the column latch; and

a variable speed system clock having an output coupled to said processing unit;

said processing unit, said variable speed system clock, said plurality of column latches, and said memory fabricated on a single substrate, said memory using a greater area of said single substrate than said processing unit, said memory further using a majority of a total area of said single substrate.

2. The microprocessor integrated circuit of claim 1 wherein said memory is dynamic random-access memory.

3. The microprocessor integrated circuit of claim 1 wherein said memory is static random-access memory.

4. A microprocessor integrated circuit comprising:

a processing unit disposed upon an integrated circuit substrate, said processing unit operating in accordance with a predefined sequence of program instructions;

a memory coupled to said processing unit and capable of storing information provided by said processing unit, said memory occupying a larger area of said integrated circuit substrate than said processing unit said memory further occupying a majority of a total area of said single substrate; and

a ring oscillator having a variable output frequency wherein the ring oscillator provides a system clock to the processing unit, the ring oscillator disposed on said integrated circuit substrate.

5. The microprocessor integrated circuit of claim 4 wherein said memory is dynamic random-access memory.

6. The microprocessor integrated circuit of claim 4 wherein said memory is static random-access memory.

7. The microprocessor integrated circuit of claim 4 wherein said memory is capable of supporting read and write operations.

8. A microprocessor integrated circuit comprising:

a processing unit having one or more interface ports for interprocessor communication, said processing unit being disposed on a single substrate;

a memory disposed upon said substrate and coupled to said processing unit, said memory occupying a greater area of said substrate than said processing unit, said memory further comprising a majority of a total area of said substrate; and

a ring oscillator having a variable output frequency wherein the ring oscillator provides a system clock to the processing unit, the ring oscillator disposed on said substrate.

9. The microprocessor integrated circuit of claim 8 wherein a first of said interface ports includes a column latch, said column latch facilitating serial communication through said first of said interface ports.

10. The microprocessor integrated circuit of claim 8 further including memory controller means coupled to said memory for performing direct memory access data transfer through said one or more interface ports.

11. A microprocessor computational system comprising:

a first processing unit disposed upon a first substrate;

a first memory disposed upon said first substrate and coupled to said first processing unit, said first memory occupying a greater area of said first substrate than said first processing unit, said memory further occupying a majority of a total area of said substrate;

a ring oscillator having a variable output frequency, wherein the ring oscillator provides a system clock to the processing unit, the ring oscillator disposed on said first substrate; and

a second processing unit coupled to said first processing unit and configured for interprocessor communication with said first processing unit.

12. The microprocessor computational system of claim 11 wherein said second processing unit and a second memory are disposed upon a second substrate, said second memory occupying a greater area of said second substrate than said second processing unit said second memory further occupying a majority of a total area of said substrate.

13. The multiprocessor computational system of claim 11 wherein said first processing unit includes an interface port for establishing said interprocessor communication between an internal register of said first processing unit and second processing unit.

* * * * *

EXHIBIT B

TO THE DECLARATION OF JEFFREY M. FISHER ISO DEFENDANTS' REPLY ISO MOTION TO DISMISS OR, IN THE ALTERNATIVE, TO TRANSFER VENUE

IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

FILED
U.S. DISTRICT COURT
EASTERN DISTRICT OF TEXAS
APR 25 2008
DAVID J. MALAND, CLERK
BY
DEPUTY________

(1) TECHNOLOGY PROPERTIES LIMITED and (2) PATRIOT SCIENTIFIC CORPORATION,

Plaintiffs,

vs.

(1) ACER, INC.
(2) ACER AMERICA CORPORATION
(3) GATEWAY, INC.

Defendants.

CASE NO 2-08 CV-173 TJW/CE

Jury Trial Demanded

COMPLAINT FOR PATENT INFRINGEMENT AND DEMAND FOR JURY TRIAL

Plaintiffs, Technology Properties Limited ("TPL") and Patriot Scientific Corporation ("Patriot"), (collectively "Plaintiffs"), allege the following in support of their Complaint for Patent Infringement and Demand for Jury Trial ("Complaint") against Defendants, Acer, Inc, ("Acer"), Acer America Corporation ("Acer America") and Gateway, Inc. ("Gateway").

PARTIES

1 Plaintiff, Technology Properties Limited ("TPL") is a corporation duly organized and existing under the laws of the State of California and maintains its principal place of business in Cupertino, California

2. Plaintiff, Patriot Scientific Corporation ("Patriot") is a corporation duly organized and existing under the laws of the State of Delaware and maintains its principal place of business in Carlsbad, California.

3. Upon information and belief, Defendant Acer, Inc. is a Taiwan corporation with its principal place of business in Taipei, Taiwan, R.O.C.

4. Upon information and belief, Defendant Acer America Corporation is a California corporation with its principal place of business in San Jose, California

5. Upon information and belief, Defendant Gateway, Inc. is a Delaware corporation with its principal place of business in Irvine, California. Gateway is a wholly-owned subsidiary of Acer

JURISDICTION

6. This Court has subject matter jurisdiction over this action pursuant to 28 U.S.C. §§ 1331, 1338(a) because this action arises under the patent laws of the United States, including 35 U.S.C. §§ 101, *et seq.* and 271, *et seq.* This Court has personal jurisdiction over Defendants because they each infringe Plaintiffs' patent by offering on their websites infringing products to their users and/or customers who reside in, or may be found in, the Eastern District of Texas. Further, each Defendant has actually transacted business with users of their websites in the Eastern District of Texas.

VENUE

7. Venue is proper in this judicial district under 28 U.S.C. §§ 1391(b) and (c) and 1400(b) because Defendants have each committed acts of infringement in this district

GENERAL ALLEGATIONS

8. On July 21, 1998, United States Patent No. 5,784,584 ("584 Patent") entitled "High Performance Microprocessor Using Instructions That Operate Within Instruction Groups" was duly and legally issued. All rights and interest in the '584 Patent are co-owned by TPL and Patriot Scientific Corporation. TPL has the sole and exclusive right and obligation to license and enforce the '584 Patent. A true and correct copy of the '584 Patent is attached hereto as Exhibit A.

COUNT 1

(Patent Infringement Against Acer, Inc.)

9. Paragraphs 1-8 of the Complaint set forth above are incorporated herein by reference.

10. Upon information and belief Defendant Acer has infringed and continues to infringe under 35 U.S.C. § 271 the '584 Patent.

11. Acer's acts of infringement have caused damage to Plaintiffs. Under 35 U.S.C. § 284, Plaintiffs are entitled to recover from Acer the damages sustained by Plaintiffs as a result of Acer's infringement of the '584 Patent. Acer's infringement of Plaintiffs' exclusive rights under the '584 Patent will continue to damage Plaintiffs' business, causing irreparable harm, for which there is no adequate remedy at law, unless enjoined by this Court under 35 U.S.C. § 283.

12. Plaintiffs allege, on information and belief, that Acer's acts of infringement were willful and deliberate.

COUNT 2

(Patent Infringement Against Acer America Corporation)

13 Paragraphs 1-8 of the Complaint set forth above are incorporated herein by reference

14. Upon information and belief Defendant Acer America has infringed and continues to infringe under 35 U S.C. § 271 the '584 Patent.

15 Acer America's acts of infringement have caused damage to Plaintiffs Under 35 U S.C § 284, Plaintiffs are entitled to recover from Acer America the damages sustained by Plaintiffs as a result of Acer America's infringement of the '584 Patent. Acer America's infringement of Plaintiffs' exclusive rights under the '584 Patent will continue to damage Plaintiffs' business, causing irreparable harm, for which there is no adequate remedy at law, unless enjoined by this Court under 35 U S C. § 283

16. Plaintiffs allege, on information and belief, that Acer America's acts of infringement were willful and deliberate.

COUNT 3

(Patent Infringement Against Gateway, Inc.)

17 Paragraphs 1-8 of the Complaint set forth above are incorporated herein by reference

18. Upon information and belief Defendant Gateway has infringed and continues to infringe under 35 U.S C. § 271 the '584 Patent

19. Gateway's acts of infringement have caused damage to Plaintiffs. Under 35 U.S.C § 284, Plaintiffs are entitled to recover from Gateway the damages sustained by Plaintiffs as a result of Gateway's infringement of the '584 Patent. Gateway's infringement of

Plaintiffs' exclusive rights under the '584 Patent will continue to damage Plaintiffs' business, causing irreparable harm, for which there is no adequate remedy at law, unless enjoined by this Court under 35 U.S.C. § 283.

20 Plaintiffs allege, on information and belief, that Gateway's acts of infringement were willful and deliberate

PRAYER FOR RELIEF

WHEREFORE, Plaintiffs respectfully request that this Court enter judgment against Defendants as follows:

A. For judgment that Defendants Acer, Inc., Acer America Corporation, and Gateway, Inc. have infringed and continue to infringe the '584 Patent;

B. For permanent injunctions under 35 U.S.C. § 283 against Defendants and their directors, officers, employees, agents, subsidiaries, parents, attorneys, and all persons acting in concert, on behalf of, in joint venture, or in partnership with Defendants from further acts of infringement;

C For damages to be paid by Defendants adequate to compensate Plaintiffs for their infringement, including interests, costs and disbursements as the Court may deem appropriate under 35 U.S.C. § 284;

D For judgment finding that Defendants' infringement was willful and deliberate, entitling Plaintiffs to increased damages under 35 U.S.C. § 284;

E. For judgment finding this to be an exceptional case against Defendants and awarding Plaintiffs attorney fees under 35 U.S.C. § 285; and,

F. For such other and further relief at law and in equity as the court may deem just and proper.

DEMAND FOR JURY TRIAL

Pursuant to the Federal Rules of Civil Procedure Rule 38, Plaintiffs hereby demand a jury trial on all issues triable by jury

Dated: April 25, 2008

Respectfully submitted,

By: [signature]

S. Calvin Capshaw
State Bar No. 03783900
Email: ccapshaw@capshawlaw.com
Elizabeth L. DeRieux
State Bar No. 05770585
Email: ederieux@capshawlaw.com
Capshaw DeRieux, LLP
1127 Judson Road, Suite 220
Longview, TX 75601
Telephone: (903) 236-9800
Facsimile: (903) 236-8787

John L. Cooper
California Bar No. 50324
Email: jcooper@fbm.com
Jeffrey M. Fisher
California Bar No. 155284
Email: jfisher@fbm.com
Helen Dutton
California Bar No. 235558
hdutton@fbm.com
Farella Braun + Martel, LLP
Russ Building
235 Montgomery Street
San Francisco, CA 94104
Telephone: (415) 954-4400
Facsimile: (415) 954-4480

ATTORNEYS FOR PLAINTIFF
TECHNOLOGY PROPERTIES LIMITED

By: Charles T Hoge by permission EAD

Robert M. Parker
State Bar No. 15498000
Email: rmparker@pbatyler.com
Robert Christopher Bunt
State Bar No. 00787165
Email: rcbunt@pbatyler.com
Parker, Bunt & Ainsworth, P.C.
100 East Ferguson, Ste 1114
Tyler, TX 75702
Telephone: (903) 531-3535
Facsimile: (903) 533-9687

Charles T. Hoge
California Bar No. 110696
Email: choge@knlh.com
Kirby Noonan Lance & Hoge, LLP
350 Tenth Avenue, Suite 1300
San Diego, CA 92101
Telephone: (619) 231-8666
Facsimile: (619) 231-9593

ATTORNEYS FOR PLAINTIFF
PATRIOT SCIENTIFIC CORPORATION

US005784584A

United States Patent [19]

Moore et al.

[11] Patent Number: 5,784,584

[45] Date of Patent: Jul. 21, 1998

[54] HIGH PERFORMANCE MICROPROCESSOR USING INSTRUCTIONS THAT OPERATE WITHIN INSTRUCTION GROUPS

[75] Inventors: **Charles H Moore**, Woodside; **Russell H. Fish, III** Mt View, both of Calif

[73] Assignee: **Patriot Scientific Corporation**, San Diego, Calif.

[21] Appl No: **484,935**

[22] Filed: **Jun 7, 1995**

Related U.S. Application Data

[62] Division of Ser No 389 334 Aug 3, 1989 Pat No 5 440 749

[51] **Int. Cl.**[6] **G06F 9/30**

[52] **U.S. Cl.** **395/376**

[58] **Field of Search** 395/376, 382, 395/384, 588, 800.23

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,967,326 10/1990 May 395/800

5,127,091 6/1992 Bonfaral et al 395/375

Primary Examiner—David Y. Eng

Attorney, Agent, or Firm—Cooley Godward LLP

[57] ABSTRACT

A high-performance microprocessor system using instruction that access operands and instructions located relative to the current instruction group rather than located relative to the current instructions, as is the convention, is disclosed herein The microprocessor system includes a central processing unit memory and a bus connecting the central processing unit and memory An instruction fetching unit, connected to the bus, is provided for fetching instruction groups from the memory for use by the central processing unit and for storage within an instruction register An instruction supplying unit operates to supply, in succession from the instruction register to the central processing unit one or more instructions from each of the instruction groups. The system further includes an instruction decoder for configuring the instruction supplying unit to select, from the instruction register, operands associated with instructions from particular instruction groups.

29 Claims, 19 Drawing Sheets

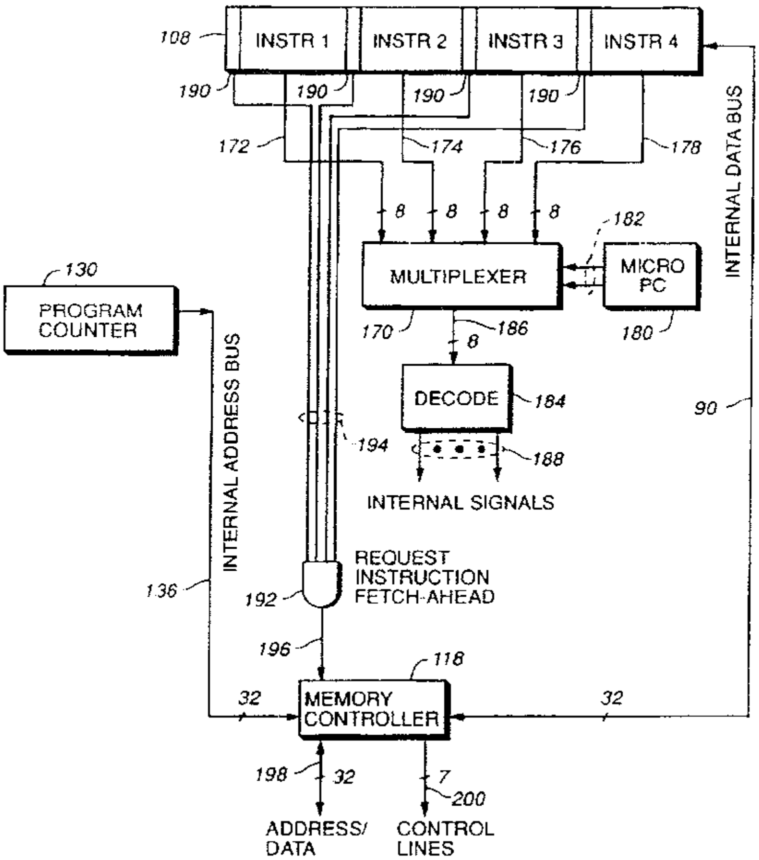
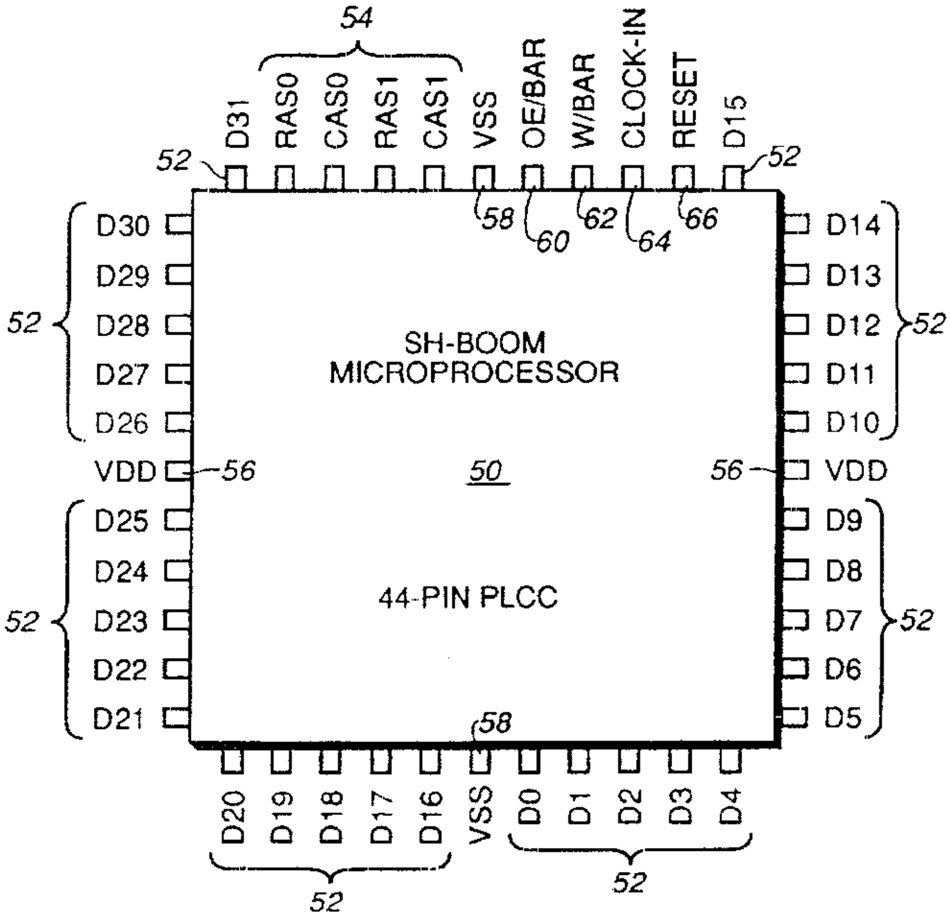


EXHIBIT A



54
D31
RAS0
CAS0
RAS1
CAS1
VSS
OE/BAR
W/BAR
CLOCK-IN
RESET
D15
52
58
60
62
64
66
D30
D29
D28
D27
D26
VDD
56
D25
D24
D23
D22
D21
D14
D13
D12
D11
D10
D9
D8
D7
D6
D5
SH-BOOM
MICROPROCESSOR
50
44-PIN PLCC
D20
D19
D18
D17
D16
D0
D1
D2
D3
D4

FIG._1

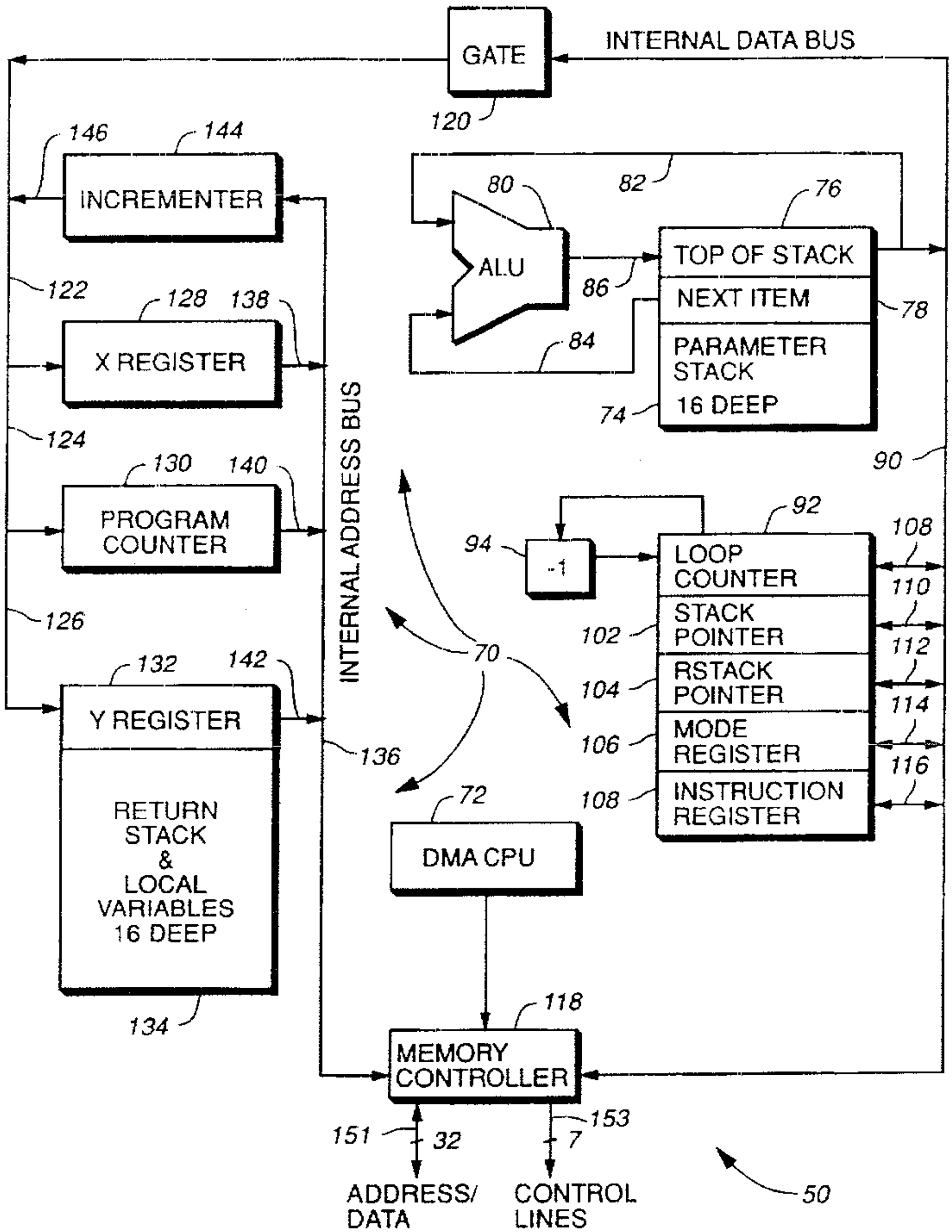
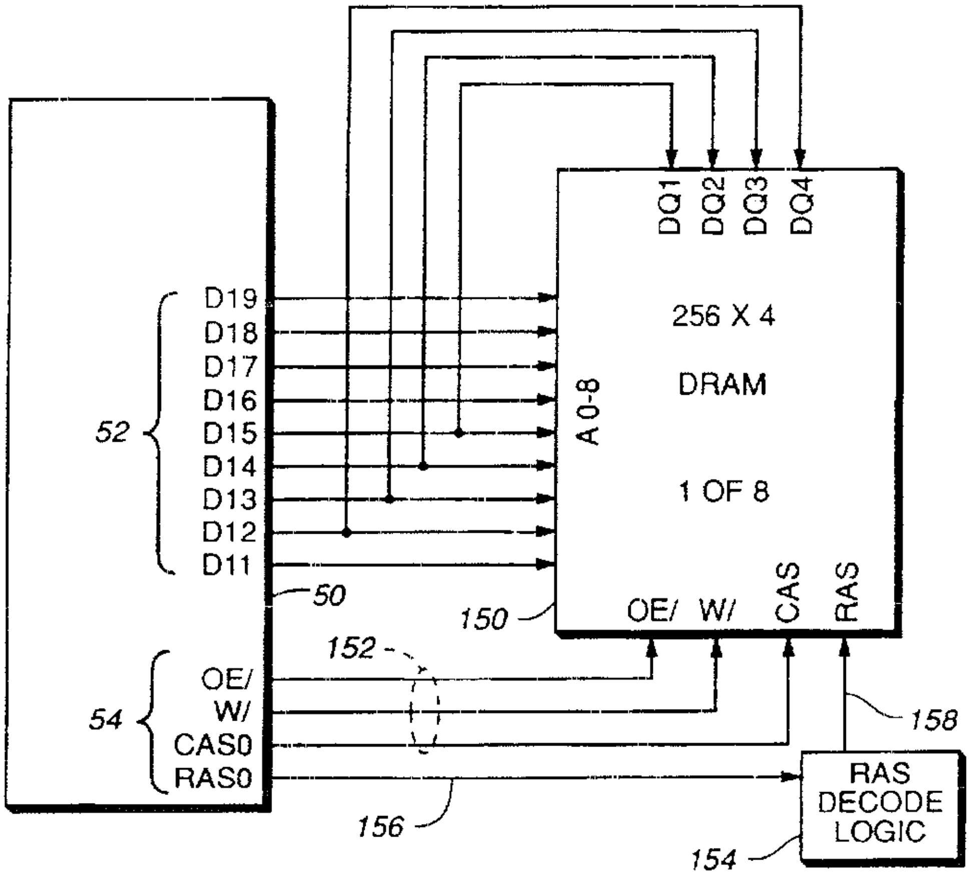
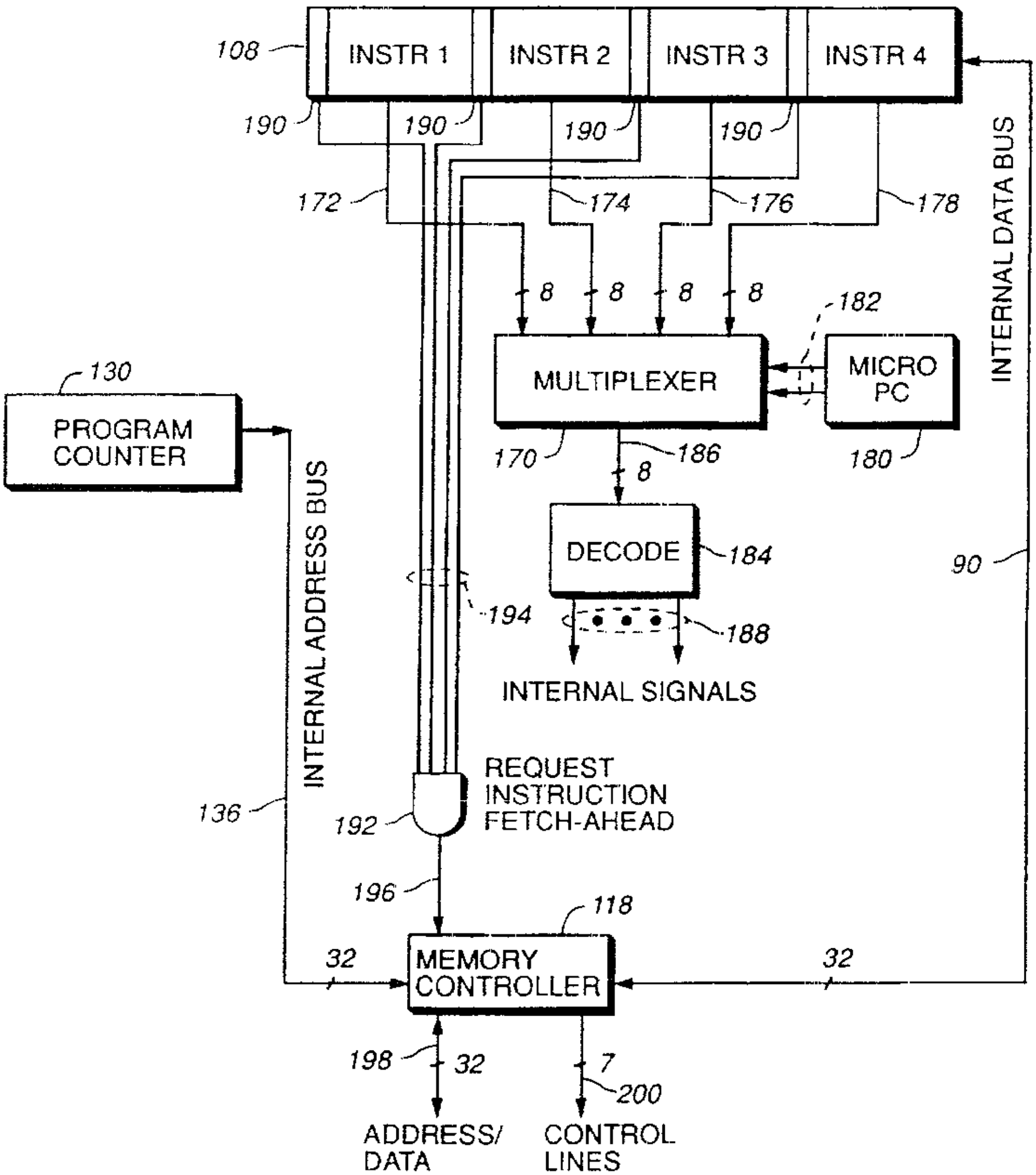

GATE
INTERNAL DATA BUS
120
146
144
INCREMENTER
80
82
76
ALU
TOP OF STACK
NEXT ITEM
PARAMETER STACK 16 DEEP
86
84
78
74
122
128
138
X REGISTER
124
130
140
PROGRAM COUNTER
INTERNAL ADDRESS BUS
90
92
94
-1
108
LOOP COUNTER
110
STACK POINTER
102
112
RSTACK POINTER
104
114
MODE REGISTER
106
116
INSTRUCTION REGISTER
108
126
132
142
70
Y REGISTER
RETURN STACK & LOCAL VARIABLES 16 DEEP
136
72
DMA CPU
134
118
MEMORY CONTROLLER
151
32
153
7
ADDRESS/ DATA
CONTROL LINES
50

FIG._2



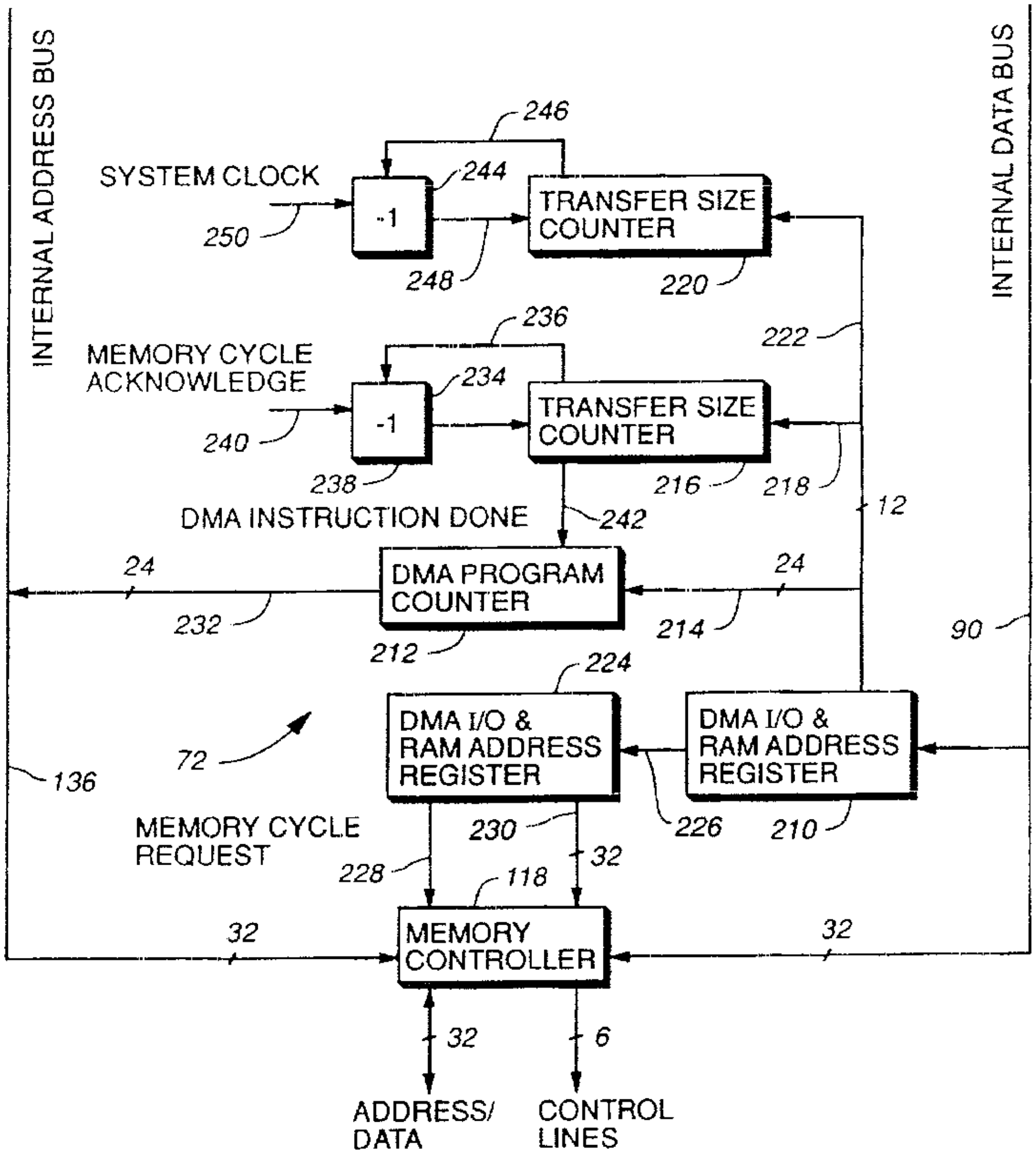
D19
D18
D17
D16
D15
D14
D13
D12
D11
52
50
OE/
W/
CAS0
RAS0
54
152
156
DQ1
DQ2
DQ3
DQ4
256 X 4
DRAM
1 OF 8
A 0-8
OE/
W/
CAS
RAS
150
158
RAS DECODE LOGIC
154

FIG._3



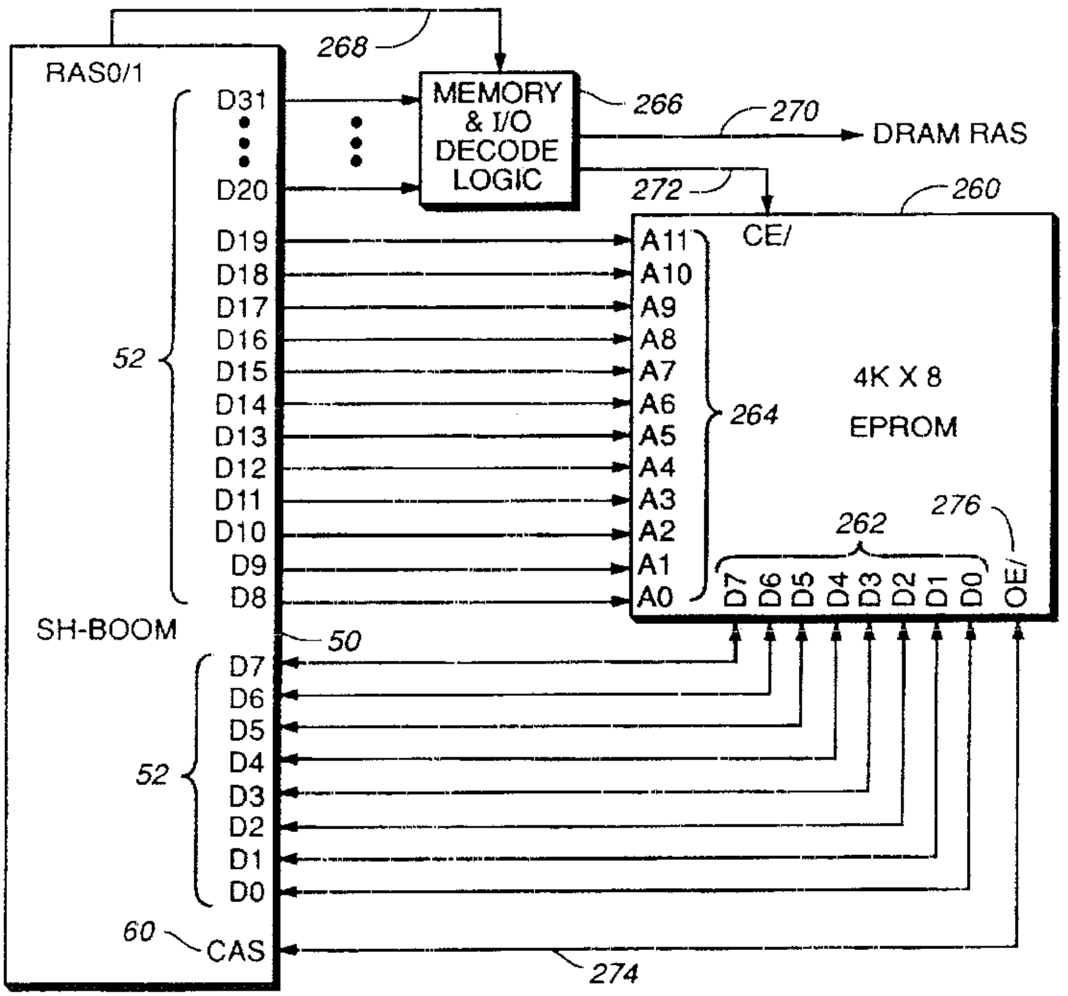
108
INSTR 1
INSTR 2
INSTR 3
INSTR 4
190
190
190
190
172
174
176
178
INTERNAL DATA BUS
8
8
8
8
182
MULTIPLEXER
MICRO PC
130
PROGRAM COUNTER
170
186
8
180
DECODE
184
90
INTERNAL ADDRESS BUS
194
188
INTERNAL SIGNALS
REQUEST INSTRUCTION FETCH-AHEAD
136
192
196
118
32
MEMORY CONTROLLER
32
198
32
7
200
ADDRESS/ DATA
CONTROL LINES

FIG._4



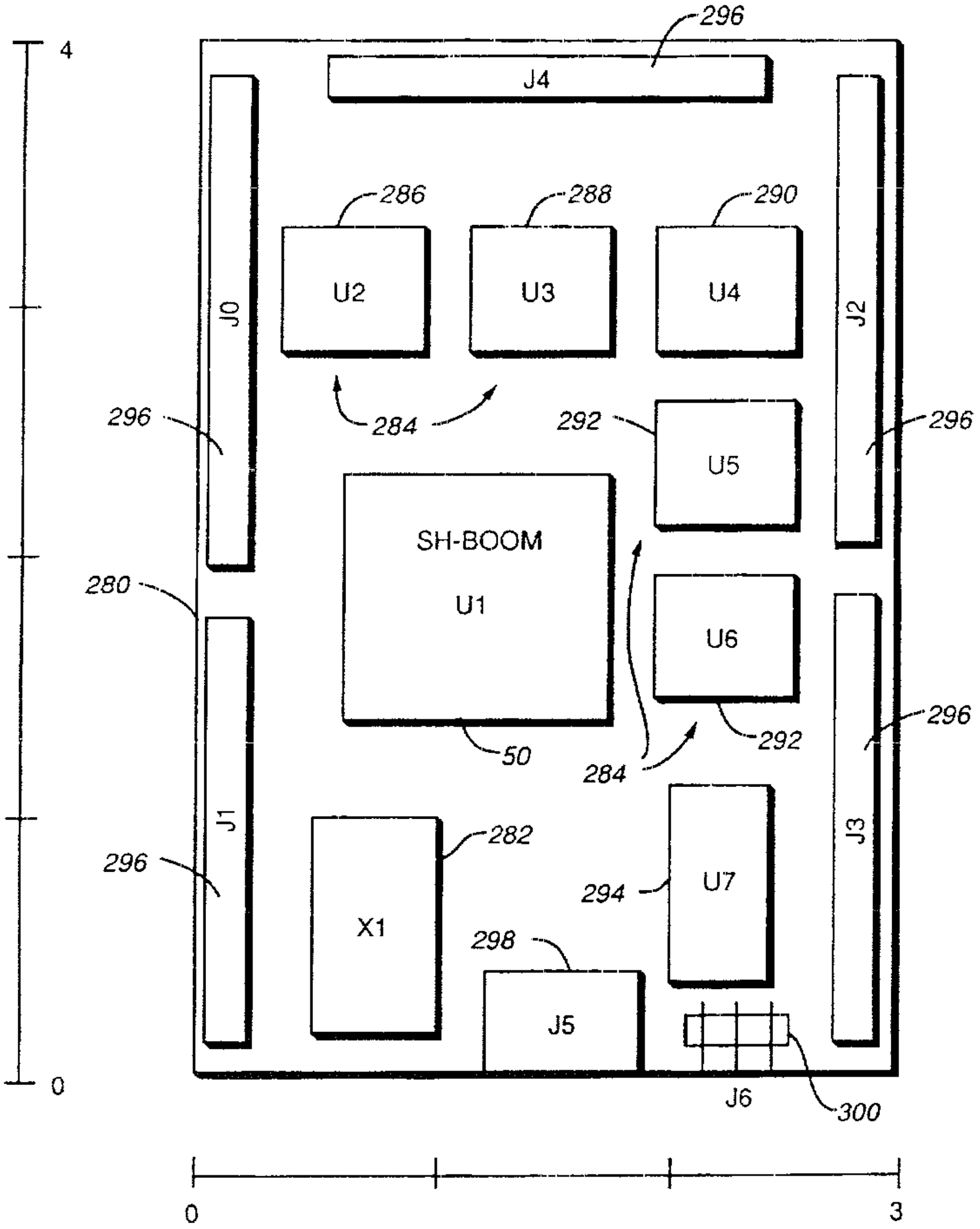
INTERNAL ADDRESS BUS
INTERNAL DATA BUS
SYSTEM CLOCK
250
-1
244
246
TRANSFER SIZE COUNTER
248
220
MEMORY CYCLE ACKNOWLEDGE
236
234
-1
TRANSFER SIZE COUNTER
240
238
216
218
222
12
DMA INSTRUCTION DONE
242
24
DMA PROGRAM COUNTER
24
232
214
212
90
224
DMA I/O & RAM ADDRESS REGISTER
DMA I/O & RAM ADDRESS REGISTER
72
136
MEMORY CYCLE REQUEST
230
226
210
228
32
118
32
MEMORY CONTROLLER
32
32
6
ADDRESS/ DATA
CONTROL LINES

FIG._5



RAS0/1
268
MEMORY & I/O DECODE LOGIC
266
270
DRAM RAS
272
260
CE/
D31
D20
D19
D18
D17
D16
D15
D14
D13
D12
D11
D10
D9
D8
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
52
264
4K X 8
EPROM
262
276
D7 D6 D5 D4 D3 D2 D1 D0 OE/
SH-BOOM
50
D7
D6
D5
D4
D3
D2
D1
D0
52
60
CAS
274

FIG._6



4
0
296
J4
J0
J2
286
288
290
U2
U3
U4
296
284
292
U5
296
SH-BOOM
U1
280
U6
296
50
284
292
J1
J3
282
X1
296
294
U7
298
J5
J6
300
0
3

FIG._7

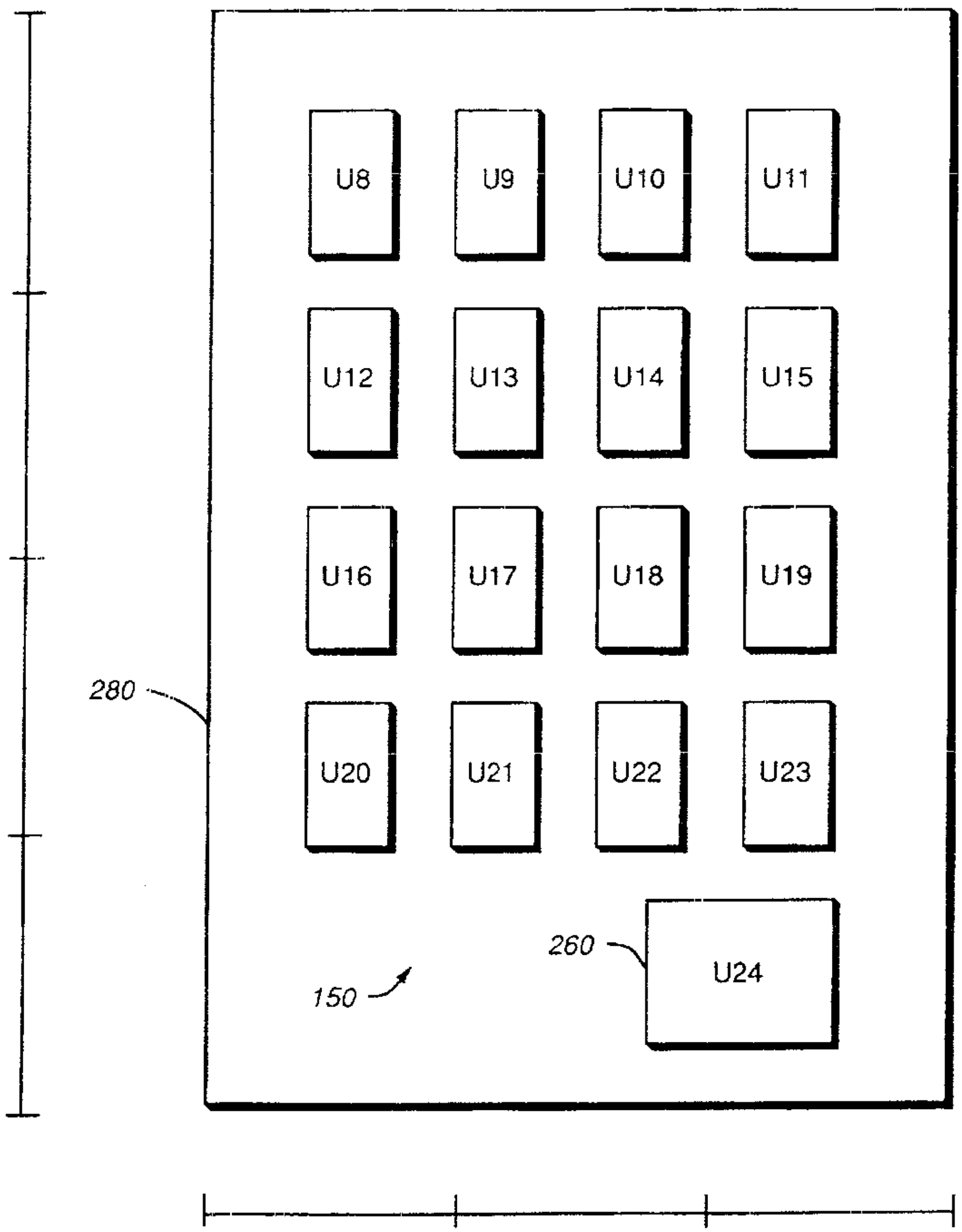

U8
U9
U10
U11
U12
U13
U14
U15
U16
U17
U18
U19
280
U20
U21
U22
U23
260
U24
150

FIG._8

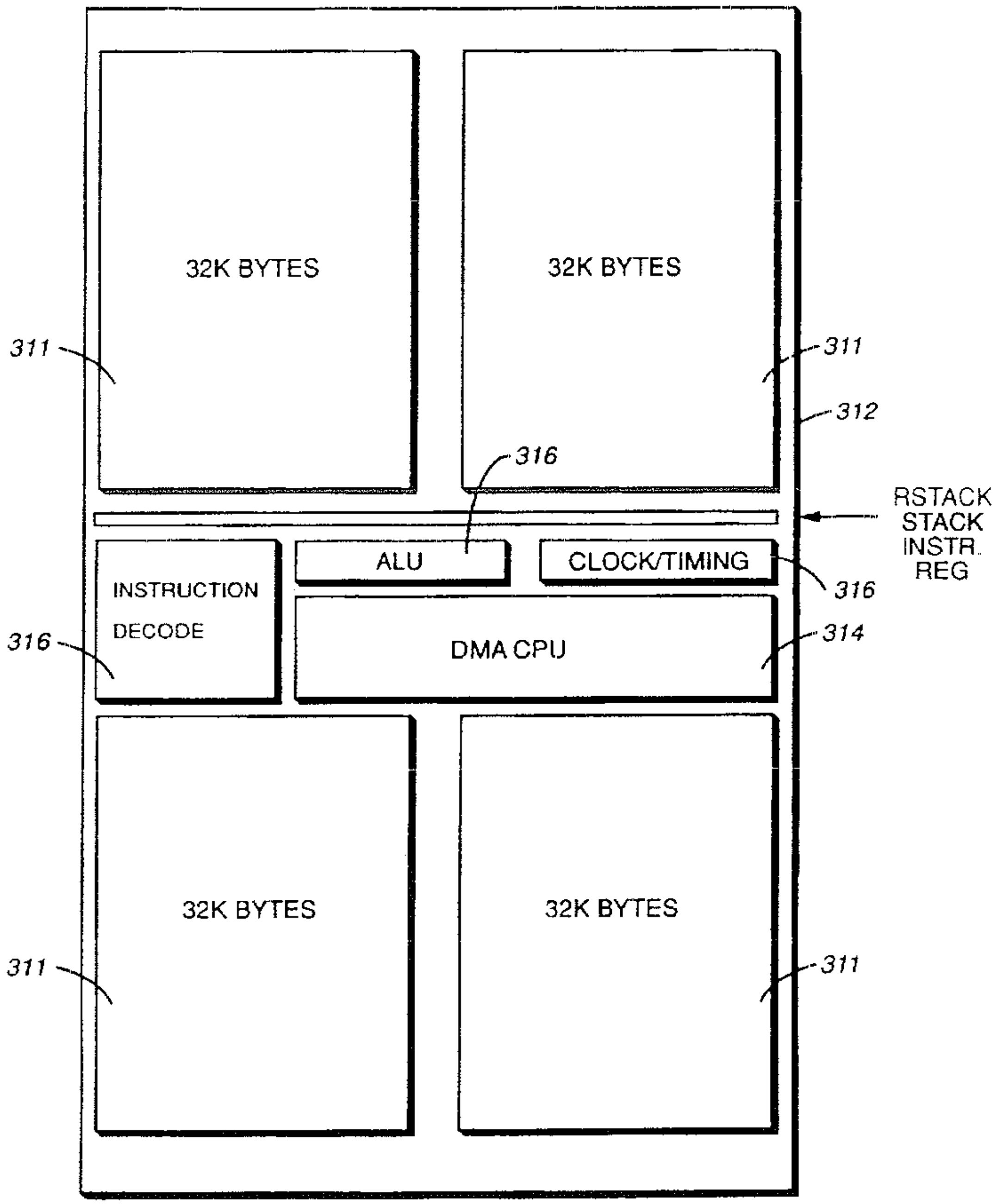

32K BYTES
32K BYTES
311
311
312
316
RSTACK
STACK
INSTR.
REG
ALU
CLOCK/TIMING
INSTRUCTION
DECODE
316
316
314
DMA CPU
32K BYTES
32K BYTES
311
311

FIG._9

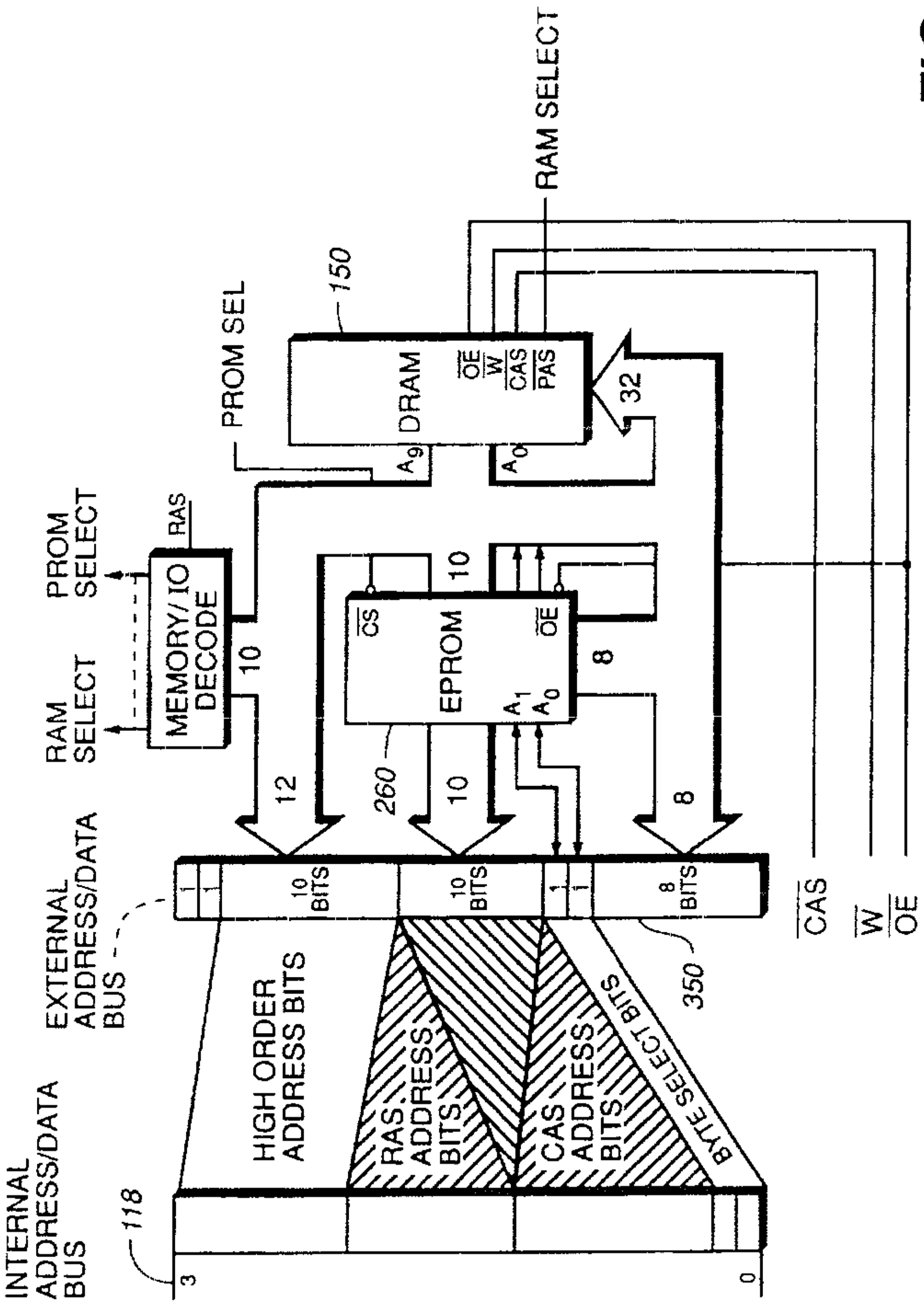
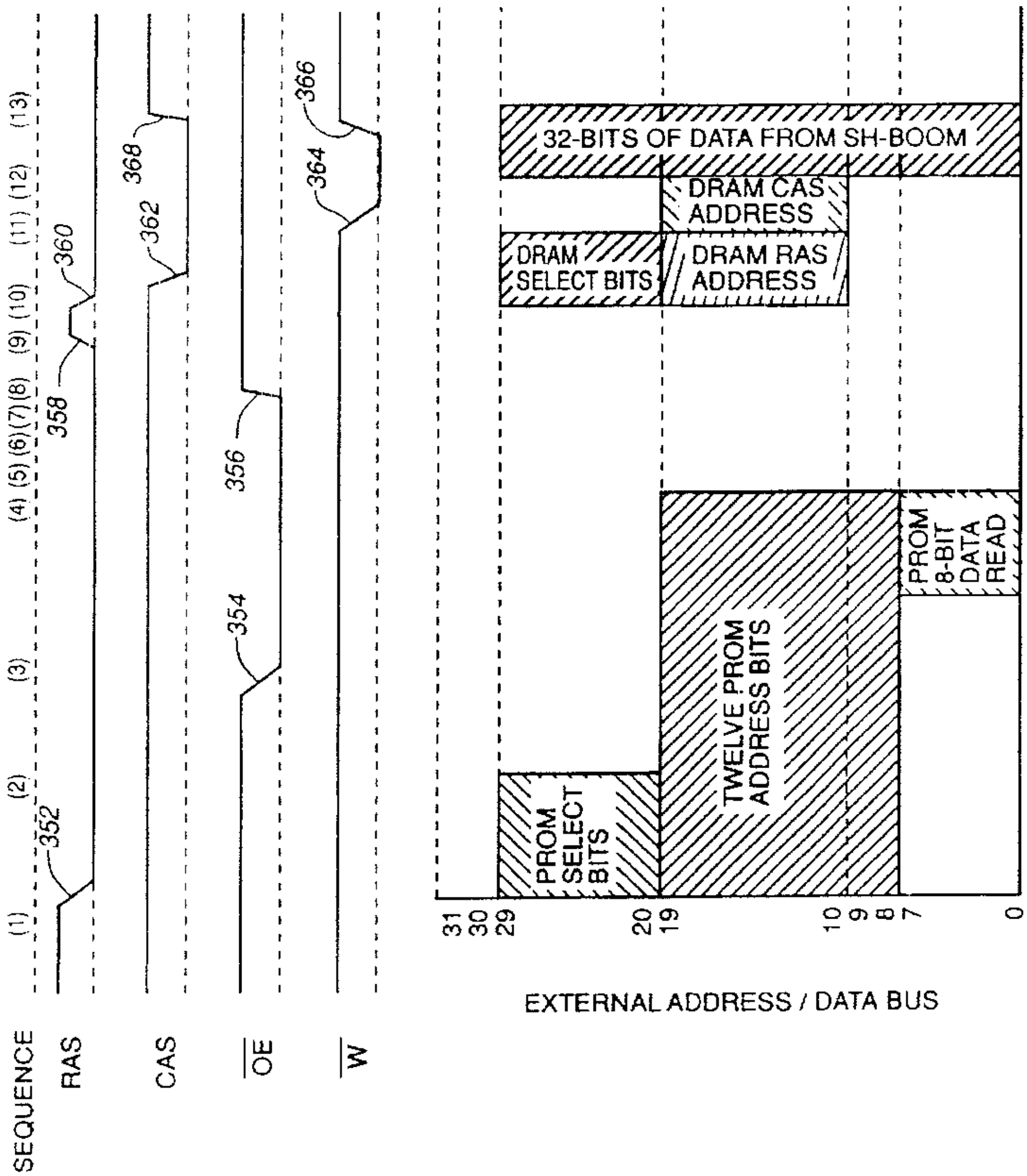

INTERNAL ADDRESS/DATA BUS
118
EXTERNAL ADDRESS/DATA BUS
RAM SELECT
PROM SELECT
MEMORY/IO DECODE
RAS
PROM SEL
150
DRAM
OE
W
CAS
PAS
A9
A0
32
RAM SELECT
EPROM
CS
OE
A1
A0
260
10
12
8
10 BITS
10 BITS
8 BITS
350
HIGH ORDER ADDRESS BITS
RAS ADDRESS BITS
CAS ADDRESS BITS
BYTE SELECT BITS
CAS
W
OE

FIG._10

FIG._11


SEQUENCE
RAS
CAS
OE
W
(1)
(2)
(3)
(4) (5) (6) (7) (8)
(9) (10)
(11) (12)
(13)
352
354
356
358
360
362
364
366
368
32-BITS OF DATA FROM SH-BOOM
DRAM CAS ADDRESS
DRAM SELECT BITS
DRAM RAS ADDRESS
TWELVE PROM ADDRESS BITS
PROM 8-BIT DATA READ
PROM SELECT BITS
31
30
29
20
19
10
9
8
7
0
EXTERNAL ADDRESS / DATA BUS

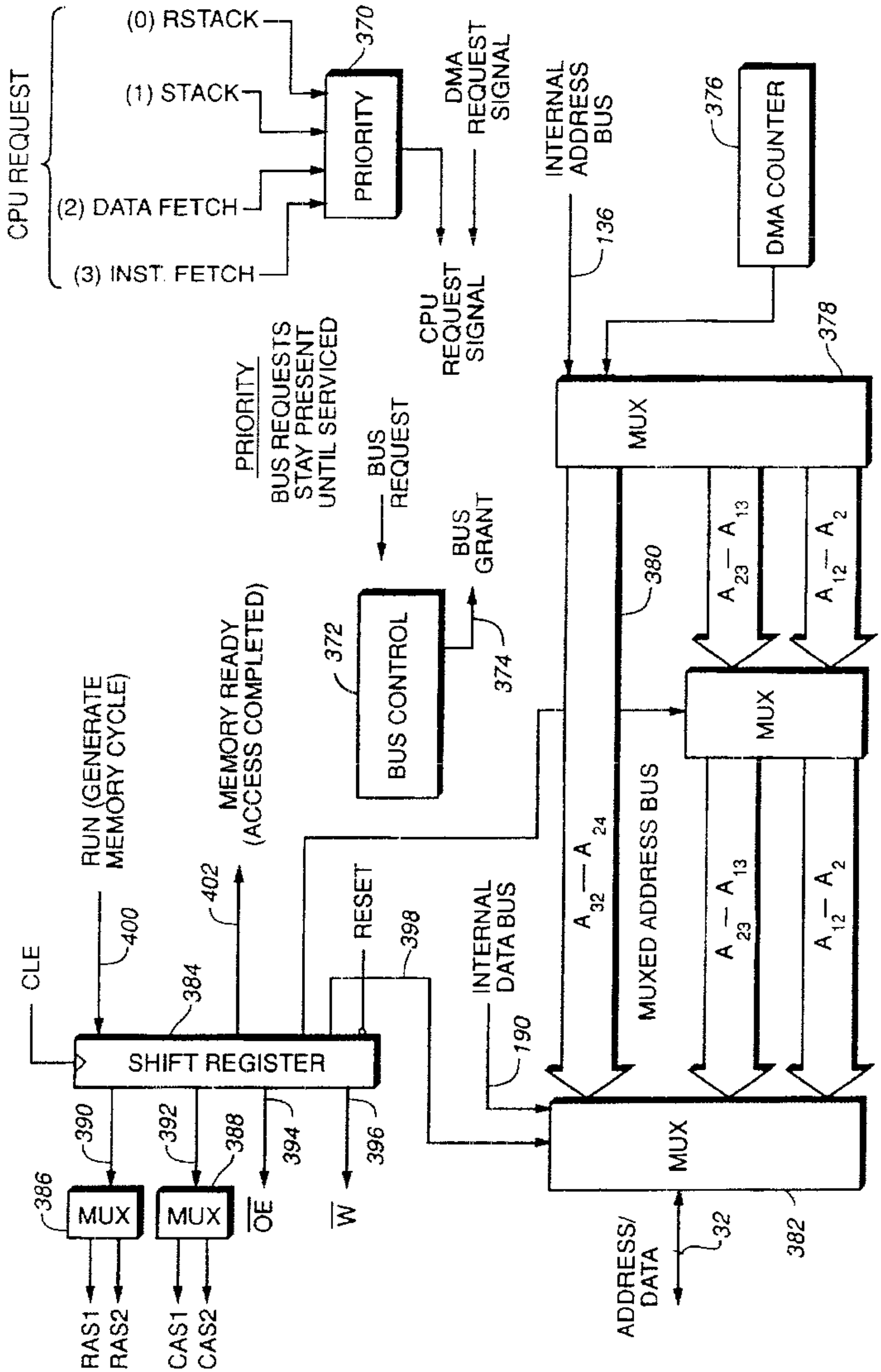

CPU REQUEST
(0) RSTACK
(1) STACK
(2) DATA FETCH
(3) INST. FETCH
PRIORITY
370
DMA REQUEST SIGNAL
CPU REQUEST SIGNAL
INTERNAL ADDRESS BUS
136
DMA COUNTER
376
378
MUX
PRIORITY
BUS REQUESTS STAY PRESENT UNTIL SERVICED
BUS REQUEST
BUS GRANT
BUS CONTROL
372
374
380
A23 — A13
A12 — A2
MUX
MEMORY READY (ACCESS COMPLETED)
RUN (GENERATE MEMORY CYCLE)
400
402
CLE
384
SHIFT REGISTER
RESET
398
INTERNAL DATA BUS
190
A32 — A24
MUXED ADDRESS BUS
A23 — A13
A12 — A2
MUX
390
392
388
394
396
386
MUX
MUX
OE
W
RAS1
RAS2
CAS1
CAS2
ADDRESS/ DATA
32
382

FIG._12

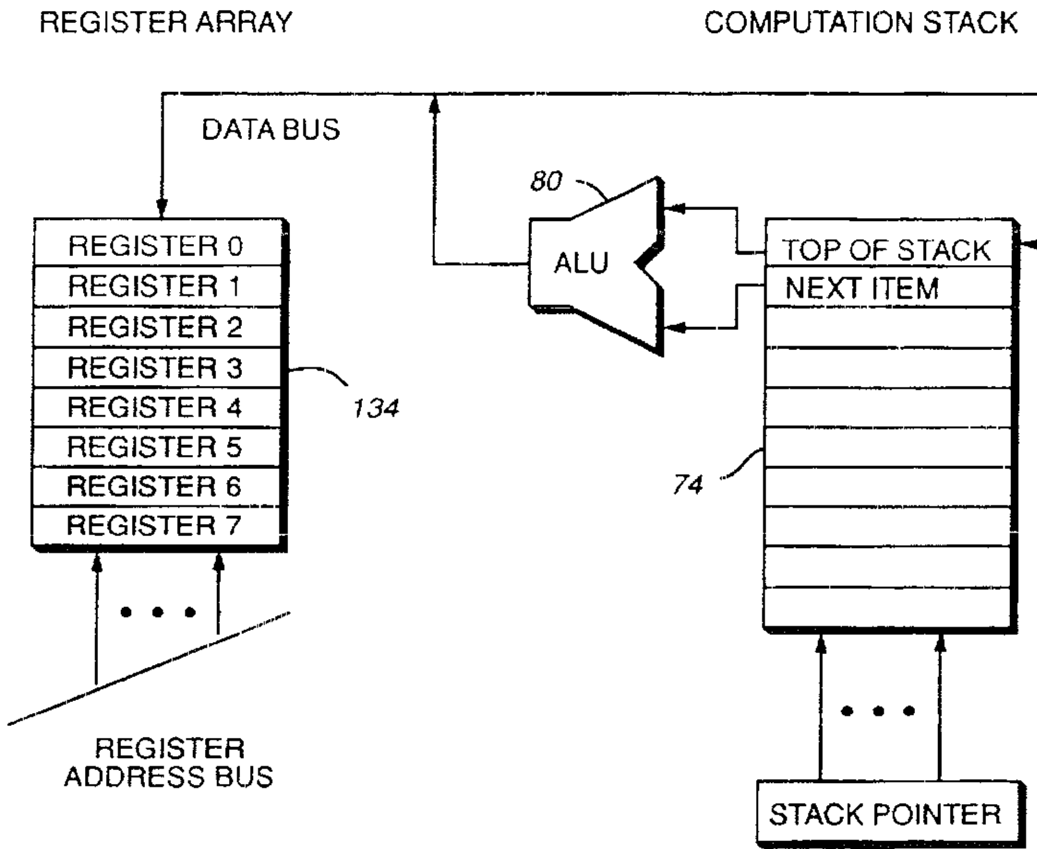

REGISTER ARRAY
COMPUTATION STACK
DATA BUS
80
ALU
REGISTER 0
REGISTER 1
REGISTER 2
REGISTER 3
REGISTER 4
REGISTER 5
REGISTER 6
REGISTER 7
134
TOP OF STACK
NEXT ITEM
74
REGISTER
ADDRESS BUS
STACK POINTER

FIG._13

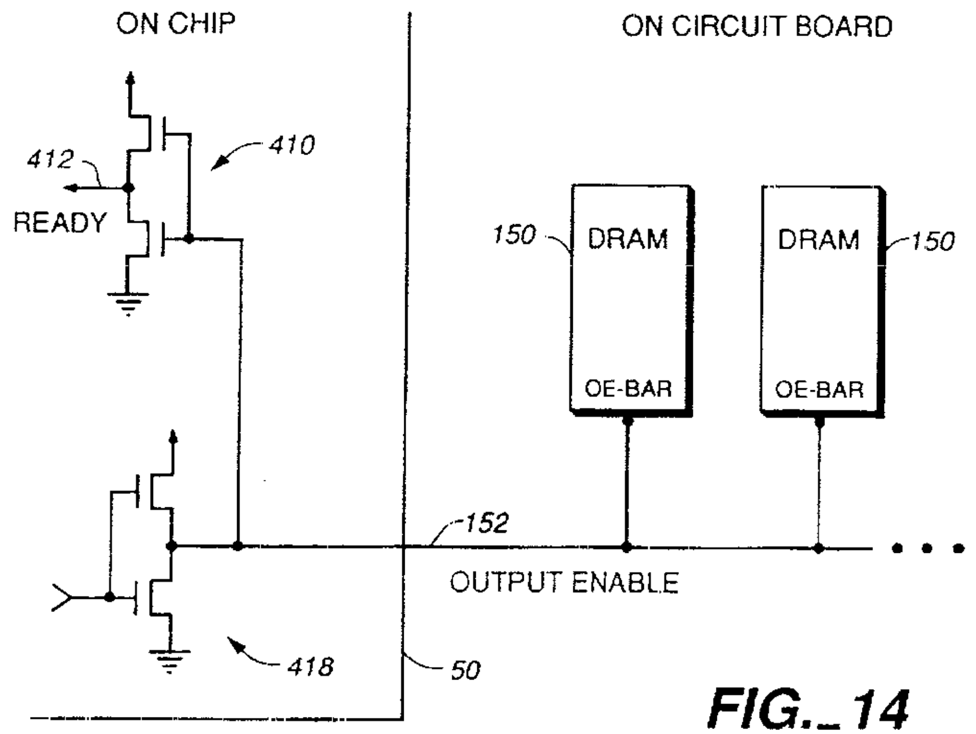

ON CHIP
ON CIRCUIT BOARD
412
410
READY
150
DRAM
DRAM
150
OE-BAR
OE-BAR
152
OUTPUT ENABLE
418
50

FIG._14

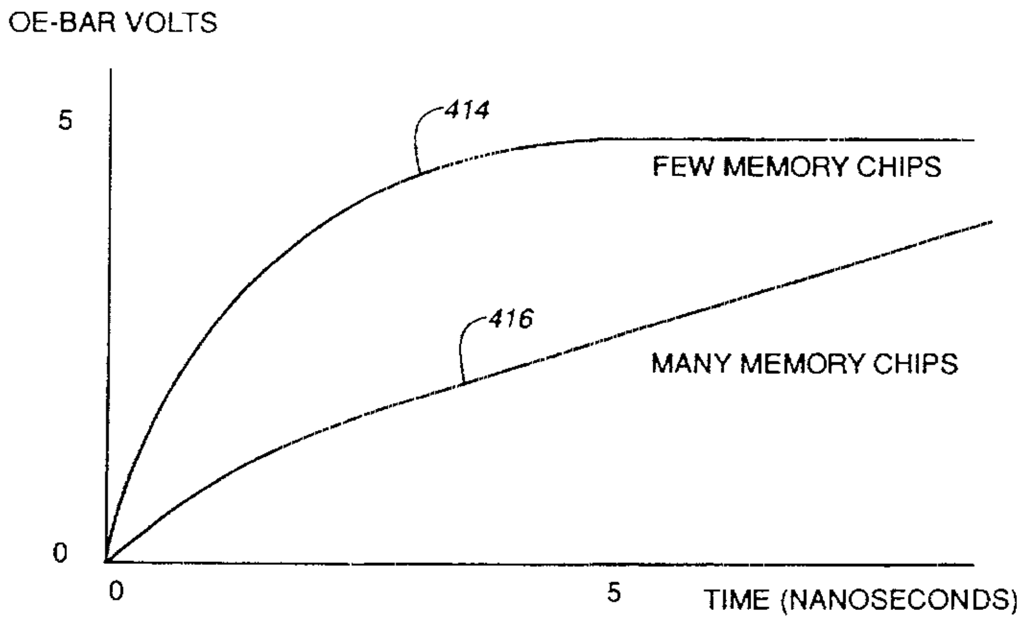

OE-BAR VOLTS
5
414
FEW MEMORY CHIPS
416
MANY MEMORY CHIPS
0
0
5
TIME (NANOSECONDS)

FIG._15

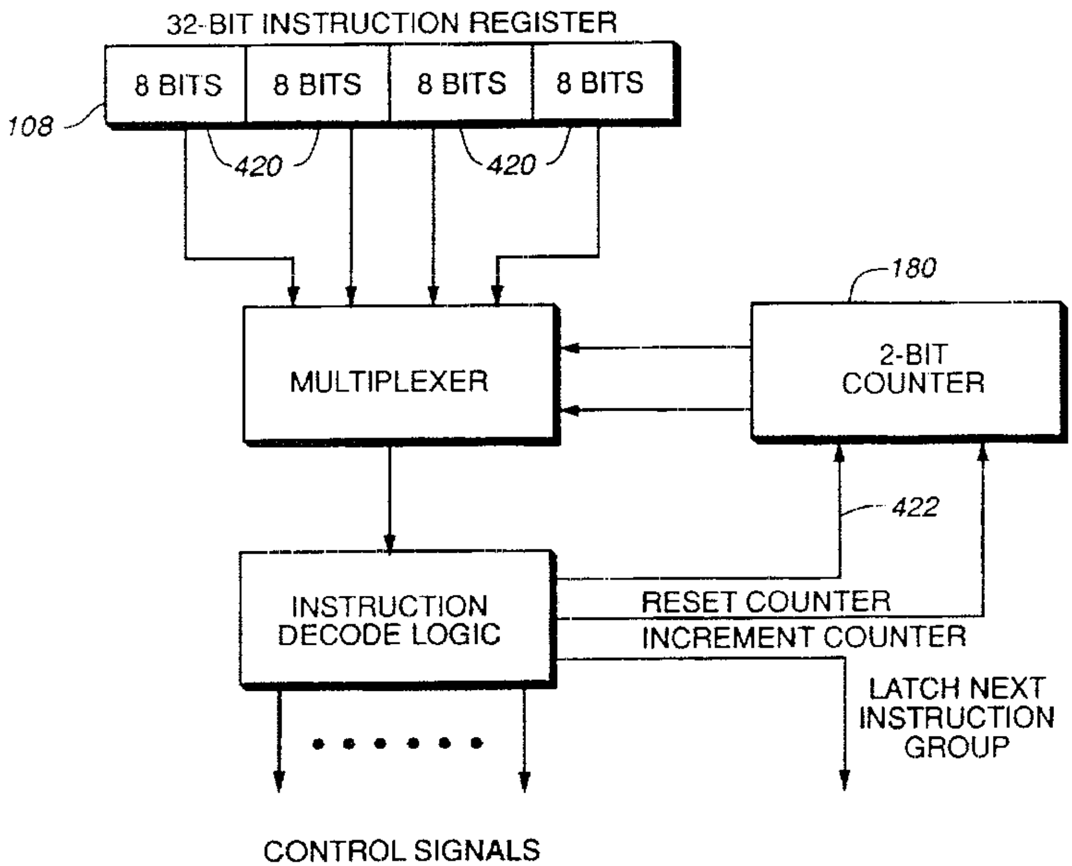

32-BIT INSTRUCTION REGISTER
8 BITS
8 BITS
8 BITS
8 BITS
108
420
420
180
MULTIPLEXER
2-BIT COUNTER
422
INSTRUCTION DECODE LOGIC
RESET COUNTER
INCREMENT COUNTER
LATCH NEXT INSTRUCTION GROUP
CONTROL SIGNALS

FIG._16

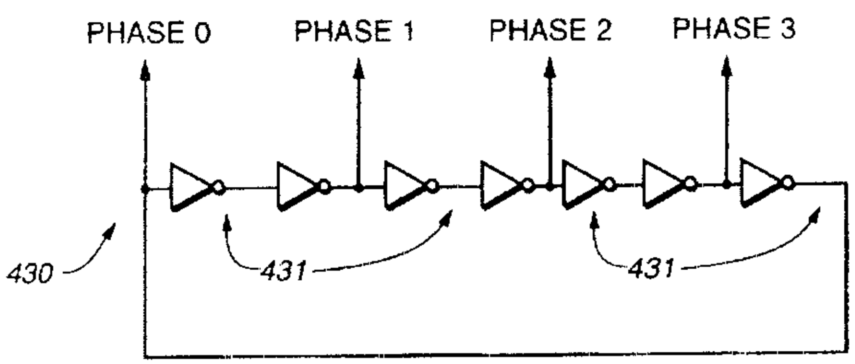

PHASE 0
PHASE 1
PHASE 2
PHASE 3
430
431
431

FIG._18

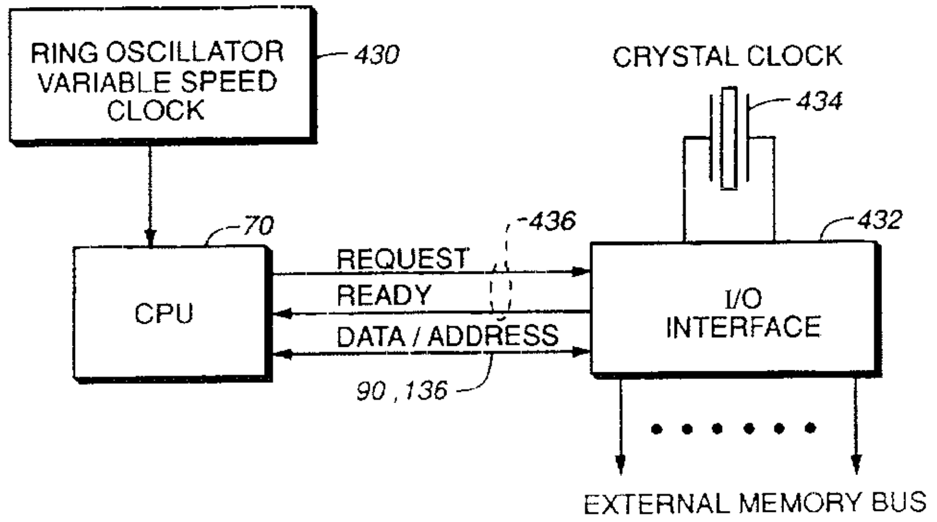

RING OSCILLATOR VARIABLE SPEED CLOCK
430
CRYSTAL CLOCK
434
70
436
432
CPU
REQUEST
READY
DATA / ADDRESS
90 , 136
I/O INTERFACE
EXTERNAL MEMORY BUS

FIG._17

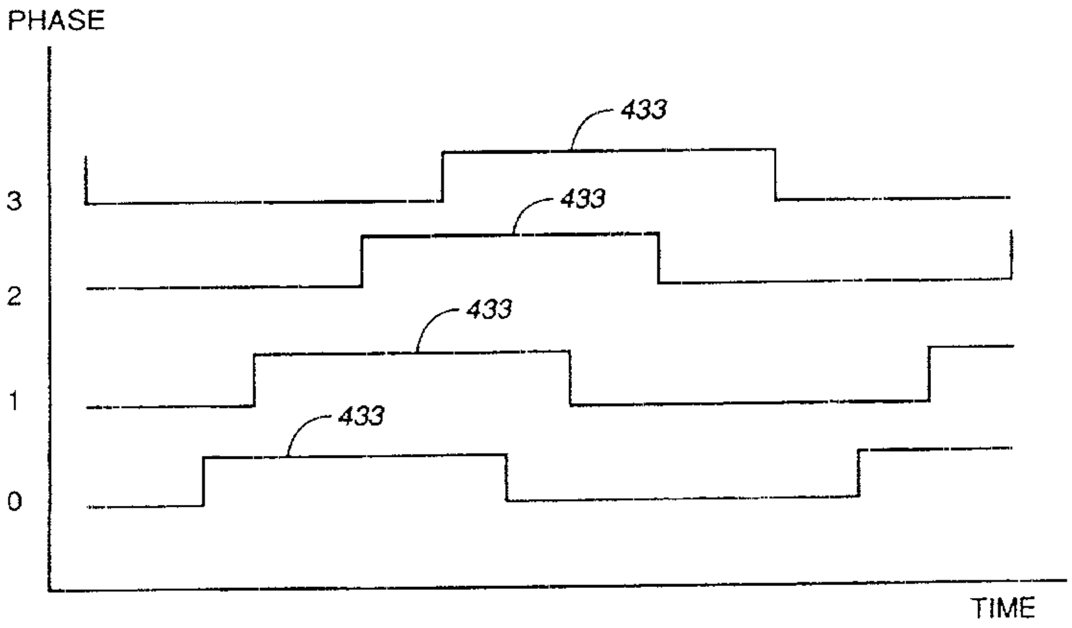

PHASE
433
3
433
2
433
1
433
0
TIME

FIG._19

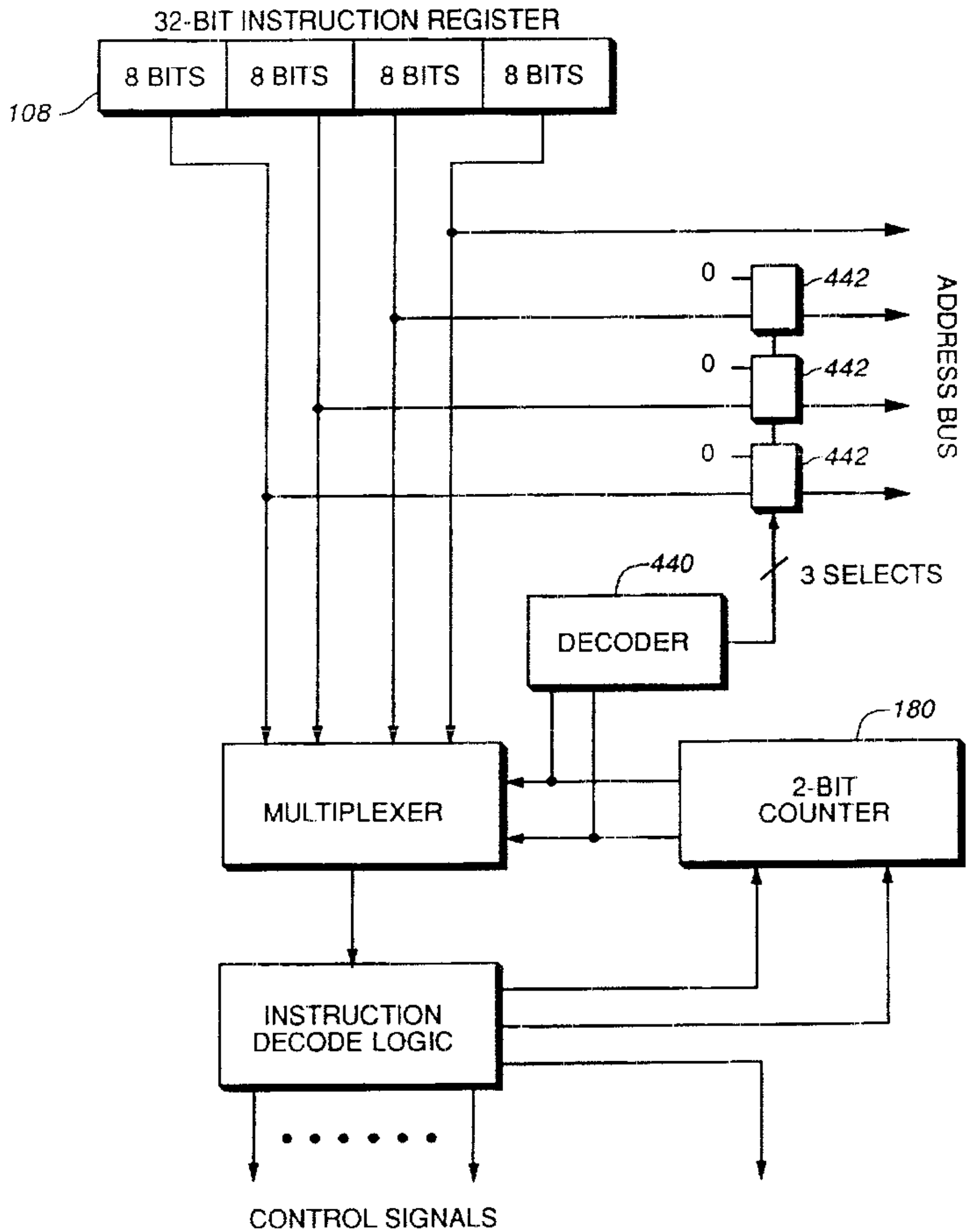
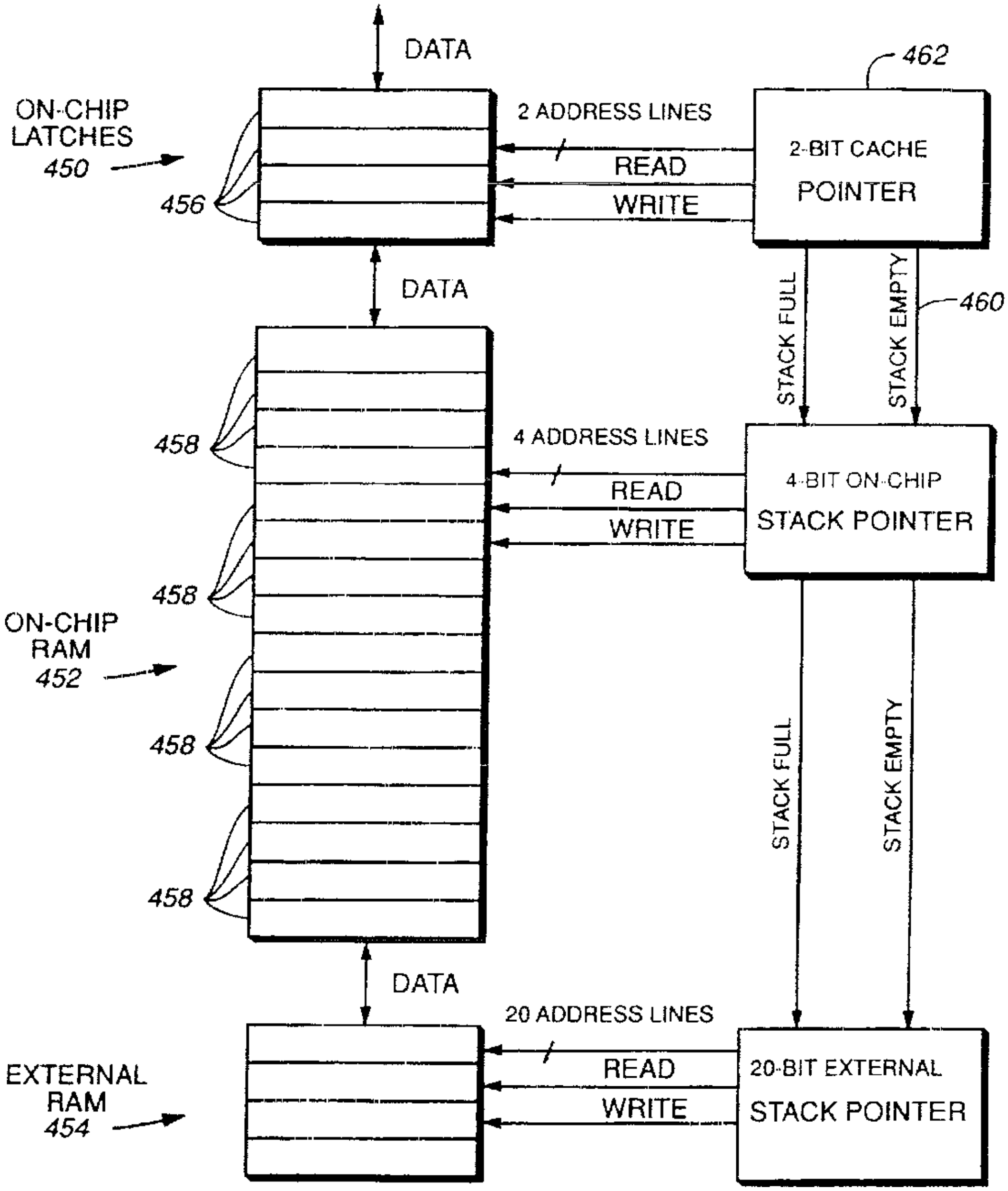
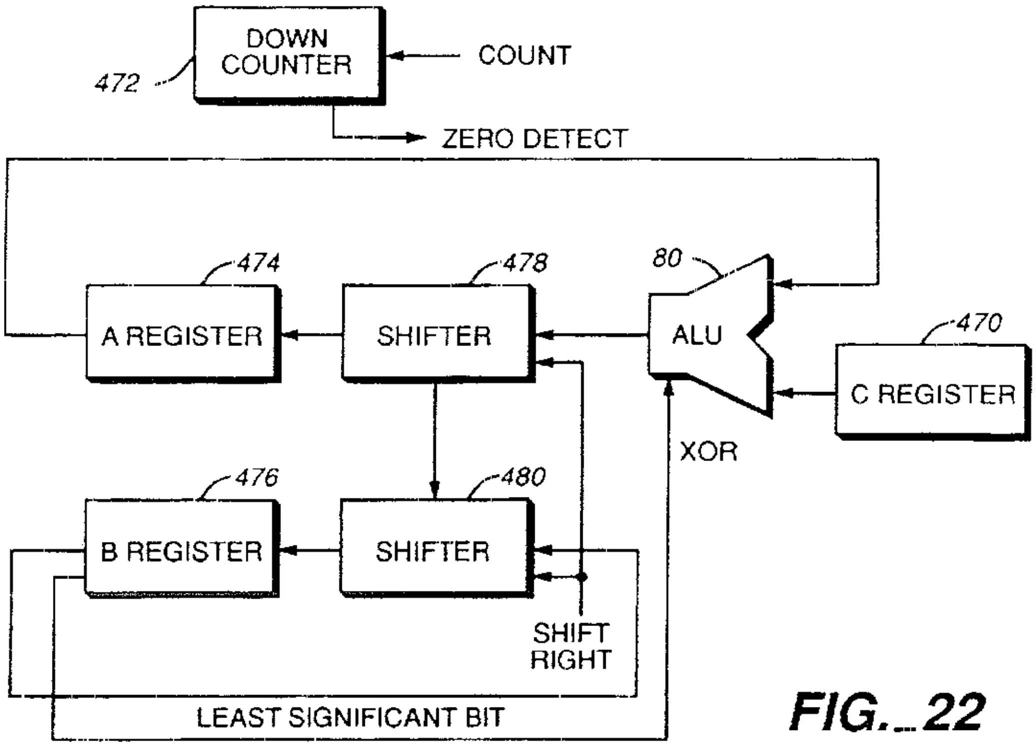

32-BIT INSTRUCTION REGISTER
8 BITS
8 BITS
8 BITS
8 BITS
108
0
442
0
442
0
442
ADDRESS BUS
440
3 SELECTS
DECODER
180
MULTIPLEXER
2-BIT
COUNTER
INSTRUCTION
DECODE LOGIC
CONTROL SIGNALS

FIG._20



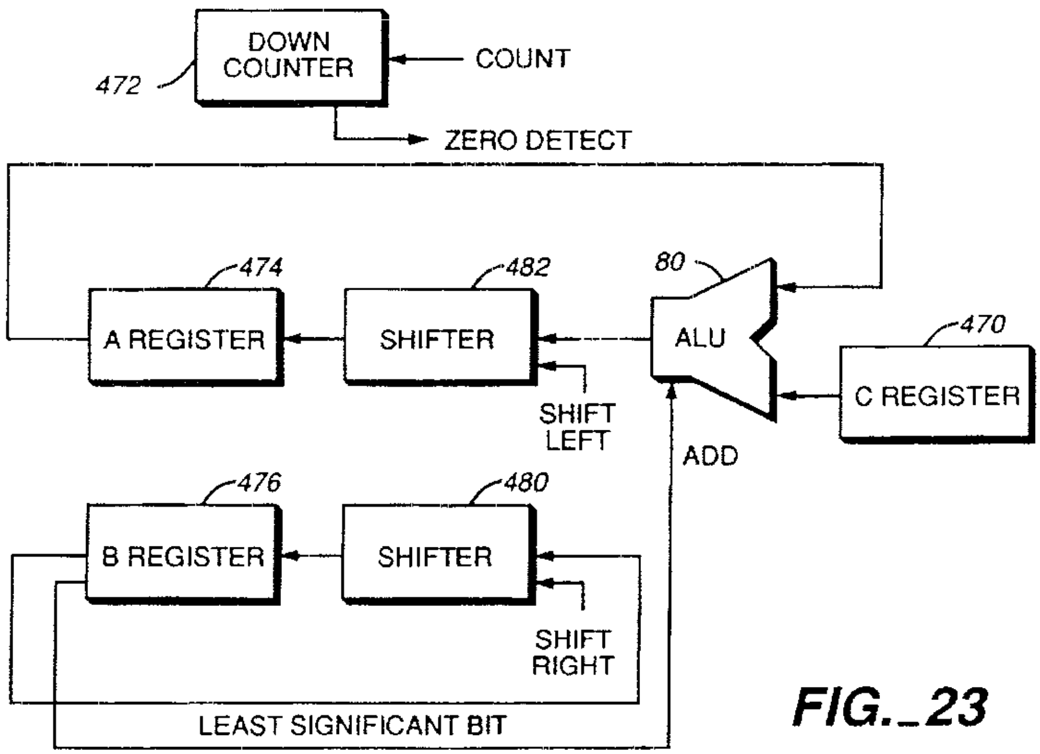
DATA
462
ON-CHIP LATCHES 450
456
2 ADDRESS LINES
READ
WRITE
2-BIT CACHE POINTER
DATA
STACK FULL
STACK EMPTY
460
458
4 ADDRESS LINES
READ
WRITE
4-BIT ON-CHIP STACK POINTER
458
ON-CHIP RAM 452
458
STACK FULL
STACK EMPTY
458
DATA
20 ADDRESS LINES
EXTERNAL RAM 454
READ
WRITE
20-BIT EXTERNAL STACK POINTER

FIG._21



DOWN COUNTER
COUNT
472
ZERO DETECT
474
478
80
A REGISTER
SHIFTER
ALU
470
C REGISTER
XOR
476
480
B REGISTER
SHIFTER
SHIFT RIGHT
LEAST SIGNIFICANT BIT

FIG._22



DOWN COUNTER
COUNT
472
ZERO DETECT
474
482
80
A REGISTER
SHIFTER
ALU
470
C REGISTER
SHIFT LEFT
ADD
476
480
B REGISTER
SHIFTER
SHIFT RIGHT
LEAST SIGNIFICANT BIT

FIG._23

HIGH PERFORMANCE MICROPROCESSOR USING INSTRUCTIONS THAT OPERATE WITHIN INSTRUCTION GROUPS

This application is a division of U S. application Ser No 07/389,334, filed Aug 3, 1989, now U.S. Pat No 5,440, 749.

BACKGROUND OF THE INVENTION

1 Field of the Invention

The present invention relates generally to a simplified, reduced instruction set computer (RISC) microprocessor. More particularly, it relates to such a microprocessor which is capable of performance levels of, for example, 20 million instructions per second (MIPS) at a price of, for example 20 dollars.

2. Description of the Prior Art

Since the invention of the microprocessor improvements in its design have taken two different approaches. In the first approach, a brute force gain in performance has been achieved through the provision of greater numbers of faster transistors in the microprocessor integrated circuit and an instruction set of increased complexity. This approach is exemplified by the Motorola 68000 and Intel 80X86 microprocessor families. The trend in this approach is to larger die sizes and packages, with hundreds of pinouts.

More recently, it has been perceived that performance gains can be achieved through comparative simplicity, both in the microprocessor integrated circuit itself and in its instruction set. This second approach provides RISC microprocessors, and is exemplified by the Sun SPARC and the Intel 8960 microprocessors. However, even with this approach as conventionally practiced, the packages for the microprocessor are large, in order to accommodate the large number of pinouts that continue to be employed. A need therefore remains for further simplification of high performance microprocessors.

With conventional high performance microprocessors fast static memories are required for direct connection to the microprocessors in order to allow memory accesses that are fast enough to keep up with the microprocessors. Slower dynamic random access memories (DRAMs) are used with such microprocessors only in a hierarchical memory arrangement, with the static memories acting as a buffer between the microprocessors and the DRAMs. The necessity to use static memories increases cost of the resulting systems.

Conventional microprocessors provide direct memory accesses (DMA) for system peripheral units through DMA controllers, which may be located on the microprocessor integrated circuit, or provided separately. Such DMA controllers can provide routine handling of DMA requests and responses, but some processing by the main central processing unit (CPU) of the microprocessor is required.

SUMMARY OF THE INVENTION

Accordingly, it is an object of this invention to provide a microprocessor with a reduced pin count and cost compared to conventional microprocessors.

It is another object of the invention to provide a high performance microprocessor that can be directly connected to DRAMs without sacrificing microprocessor speed.

It is a further object of the invention to provide a high performance microprocessor in which DMA does not require use of the main CPU during DMA requests and responses and which provides very rapid DMA response with predictable response times.

The attainment of these and related objects may be achieved through use of the novel high performance, low cost microprocessor herein disclosed. In accordance with one aspect of the invention, a microprocessor system in accordance with this invention has a central processing unit, a memory and a bus connecting the central processing unit to the memory. Instruction fetching means are connected to the bus to fetch instruction groups via the bus from the memory. Each of the instruction groups include at least one instruction that accesses operands or instructions or both. The operands and instructions are located relative to the instruction groups. An instruction register receives a first of the instruction groups from the instruction fetching means. The first of the instruction groups include one or more sequential instructions. Instruction supplying means supplies, in succession from the instruction register, the one or more sequential instructions of the first of the instruction groups to the central processing unit. An instruction decoding means configures the instruction supplying means to select from the instruction register an operand associated with one of the instructions from the first of the instruction groups.

In accordance with another aspect of the invention, the microprocessor has a central processing unit and an instruction register operatively coupled to the central processing unit. An instruction fetching means provides instruction groups to the instruction register wherein certain of the instruction groups include one or more operands or sequential instructions or both. The one or more sequential instructions including at least one instruction that accesses operands or instructions or both being located relative to the instruction groups. An instruction supplying means successively couples the one or more sequential instructions of the certain of the instruction groups to the central processing unit. An instruction decoding means configures the instruction supplying means to select operands from the instruction register associated with particular ones of the sequential instructions.

In another aspect of the invention, the microprocessor system includes a central processing unit, memory, and an instruction register. A method provides instructions from the instruction register to the central processing unit and comprises the steps of:

- providing instruction groups to the instruction register from the memory wherein certain of the instruction groups include one or more operands or sequential instructions or both;
- supplying, in succession from the instruction register, the one or more sequential instructions of the certain of the instruction groups to the central processing unit; and
- selecting an operand from the one of the instruction groups for use by the central processing unit.

The attainment of the foregoing and related objects, advantages and features of the invention should be more readily apparent to those skilled in the art, after review of the following more detailed description of the invention, taken together with the drawings, in which:

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is an external, plan view of an integrated circuit package incorporating a microprocessor in accordance with the invention.

FIG. 2 is a block diagram of a microprocessor in accordance with the invention.

FIG. 3 is a block diagram of a portion of a data processing system incorporating the microprocessor of FIGS. 1 and 2.

FIG. 4 is a more detailed block diagram of a portion of the microprocessor shown in FIG. 2.

FIG. 5 is a more detailed block diagram of another portion of the microprocessor shown in FIG. 2.

FIG. 6 is a block diagram of another portion of the data processing system shown in part in FIG. 3 and incorporating the microprocessor of FIGS. 1–2 and 4–5.

FIGS. 7 and 8 are layout diagrams for the data processing system shown in part in FIGS. 3 and 6.

FIG. 9 is a layout diagram of a second embodiment of a microprocessor in accordance with the invention in a data processing system on a single integrated circuit.

FIG. 10 is a more detailed block diagram of a portion of the data processing system of FIGS. 7 and 8.

FIG. 11 is a timing diagram useful for understanding operation of the system portion shown in FIG. 12.

FIG. 12 is another more detailed block diagram of a further portion of the data processing system of FIGS. 7 and 8.

FIG. 13 is a more detailed block diagram of a portion of the microprocessor shown in FIG. 2.

FIG. 14 is a more detailed block and schematic diagram of a portion of the system shown in FIGS. 3 and 7–8.

FIG. 15 is a graph useful for understanding operation of the system portion shown in FIG. 14.

FIG. 16 is a more detailed block diagram showing part of the system portion shown in FIG. 4.

FIG. 17 is a more detailed block diagram of a portion of the microprocessor shown in FIG. 2.

FIG. 18 is a more detailed block diagram of part of the microprocessor portion shown in FIG. 17.

FIG. 19 is a set of waveform diagrams useful for understanding operation of the part of the microprocessor portion shown in FIG. 18.

FIG. 20 is a more detailed block diagram showing another part of the system portion shown in FIG. 4.

FIG. 21 is a more detailed block diagram showing another part of the system portion shown in FIG. 4.

FIGS. 22 and 23 are more detailed block diagrams showing another part of the system portion shown in FIG. 4.

DETAILED DESCRIPTION OF THE INVENTION

OVERVIEW

The microprocessor of this invention is desirably implemented as a 32-bit microprocessor optimized for:

HIGH EXECUTION SPEED, and

LOW SYSTEM COST.

In this embodiment, the microprocessor can be thought of as 20 MIPS for 20 dollars. Important distinguishing features of the microprocessor are:

Uses low-cost commodity DYNAMIC RAMS to run 20 MIPS

4 instruction fetch per memory cycle

On-chip fast page-mode memory management

Runs fast without external cache

Requires few interfacing chips

Crams 32-bit CPU in 44 pin SOJ package

The instruction set is organized so that most operations can be specified with 8-bit instructions. Two positive products of this philosophy are:

Programs are smaller.

Programs can execute much faster.

The bottleneck in most computer systems is the memory bus. The bus is used to fetch instructions and fetch and store data. The ability to fetch four instructions in a single memory bus cycle significantly increases the bus availability to handle data.

Turning now to the drawings, more particularly to FIG. 1, there is shown a packaged 32-bit microprocessor **50** in a 44-pin plastic leadless chip carrier, shown approximately 100 times its actual size of about 0.8 inch on a side. The fact that the microprocessor **50** is provided as a 44-pin package represents a substantial departure from typical microprocessor packages, which usually have about 200 input/output (I/O) pins. The microprocessor **50** is rated at 20 million instructions per second (MIPS). Address and data lines **52**, also labelled D0–D31, are shared for addresses and data without speed penalty as a result of the manner in which the microprocessor **50** operates, as will be explained below.

DYNAMIC RAM

In addition to the low cost 44-pin package, another unusual aspect of the high performance microprocessor **50** is that it operates directly with dynamic random access memories (DRAMs), as shown by row address strobe (RAS) and column address strobe (CAS) I/O pins **54**. The other I/O pins for the microprocessor **50** include V_{DD} pins **56**, V_{SS} pins **58**, output enable pin **60**, write pin **62**, clock pin **64** and reset pin **66**.

All high speed computers require high speed and expensive memory to keep up. The highest speed static RAM memories cost as much as ten times as much as slower dynamic RAMs. This microprocessor has been optimized to use low-cost dynamic RAM in high-speed page-mode. Page-mode dynamic RAMs offer static RAM performance without the cost penalty. For example, low-cost 85 nsec. dynamic RAMs access at 25 nsec when operated in fast page-mode. Integrated fast page-mode control on the microprocessor chip simplifies system interfacing and results in a faster system.

Details of the microprocessor **50** are shown in FIG. 2. The microprocessor **50** includes a main central processing unit (CPU) **70** and a separate direct memory access (DMA) CPU **72** in a single integrated circuit making up the microprocessor **50**. The main CPU **70** has a first **16** deep push down stack **74**, which has a top item register **76** and a next item register **78**, respectively connected to provide inputs to an arithmetic logic unit (ALU) **80** by lines **82** and **84**. An output of the ALU **80** is connected to the top item register **76** by line **86**. The output of the top item register at **82** is also connected by line **88** to an internal data bus **90**.

A loop counter **92** is connected to a decrementer **94** by lines **96** and **98**. The loop counter **92** is bidirectionally connected to the internal data bus **90** by line **100**. Stack pointer **102**, return stack pointer **104**, mode register **106** and instruction register **108** are also connected to the internal data bus **90** by lines **110**, **112**, **114** and **116**, respectively. The internal data bus **90** is connected to memory controller **118** and to gate **120**. The gate **120** provides inputs on lines **122**, **124**, and **126** to X register **128**, program counter **130** and Y register **132** of return push down stack **134**. The X register **128**, program counter **130** and Y register **132** provide outputs to internal address bus **136** on lines **138**, **140** and **142**. The internal address bus provides inputs to the memory controller **118** and to an incrementer **144**. The incrementer **144** provides inputs to the X register, program counter and Y register via lines **146**, **122**, **124** and **126**. The DMA CPU **72** provides inputs to the memory controller **118** on line **148**.

The memory controller **118** is connected to a RAM (not shown) by address/data bus **151** and control lines **153**

FIG. **2** shows that the microprocessor **50** has a simple architecture. Prior art RISC microprocessors are substantially more complex in design. For example the SPARC RISC microprocessor has three times the gates of the microprocessor **50**. and the Intel 8960 RISC microprocessor has 20 times the gates of the microprocessor **50**. The speed of this microprocessor is in substantial part due to this simplicity. The architecture incorporates push down stacks and register write to achieve this simplicity.

The microprocessor **50** incorporates an I/O that has been tuned to make heavy use of resources provided on the integrated circuit chip. On chip latches allow use of the same I/O circuits to handle three different things: column addressing. row addressing and data. with a slight to non-existent speed penalty. This triple bus multiplexing results in fewer buffers to expand. fewer interconnection lines. fewer I/O pins and fewer internal buffers.

The provision of on-chip DRAM control gives a performance equal to that obtained with the use of static RAMs. As a result, memory is provided at ¼ the system cost of static RAM used in most RISC systems.

The microprocessor **50** fetches 4 instructions per memory cycle; the instructions are in an 8-bit format. and this is a 32-bit microprocessor. System speed is therefore 4 times the memory bus bandwidth. This ability enables the microprocessor to break the Von Neumann bottleneck of the speed of getting the next instruction. This mode of operation is possible because of the use of a push down stack and register array. The push down stack allows the use of implied addresses rather than the prior art technique of explicit addresses for two sources and a destination.

Most instructions execute in 20 nanoseconds in the microprocessor **50**. The microprocessor can therefore execute instructions at **50** peak MIPS without pipeline delays. This is a function of the small number of gates in the microprocessor **50** and the high degree of parallelism in the architecture of the microprocessor.

FIG. **3** shows how column and row addresses are multiplexed on lines D8–D14 of the microprocessor **50** for addressing DRAM **150** from I/O pins **52**. The DRAM **150** is one of eight. but only one DRAM **150** has been shown for clarity. As shown. the lines D11–D18 are respectively connected to row address inputs A0–A8 of the DRAM 15C. Additionally. lines D12–D15 are connected to the data inputs DQ1–DQ4 of the DRAM **150**. The output enable. write and column address strobe pins **54** are respectively connected to the output enable. write and column address strobe inputs of the DRAM **150** by lines **152**. The row address strobe pin **54** is connected through row address strobe decode logic **154** to the row address strobe input of the DRAM **150** by lines **156** and **158**

D0–D7 pins **52** (FIG. **1**) are idle when the microprocessor **50** is outputting multiplexed row and column addresses on D11–D18 pins **52**. The D0–D7 pins **52** can therefore simultaneously be used for I/O when right justified I/O is desired. Simultaneous addressing and I/O can therefore be carried out.

FIG. **4** shows how the microprocessor **50** is able to achieve performance equal to the use of static RAMS with DRAMs through multiple instruction fetch in a single clock cycle and instruction fetch-ahead. Instruction register **108** receives four 8-bit byte instruction words **1–4** on 32-bit internal data bus **90**. The four instruction byte **1–4** locations of the instruction register **108** are connected to multiplexer **170** by busses **172**. **174**. **176** and **178**. respectively. A microprogram counter **180** is connected to the multiplexer **170** by lines **182**. The multiplexer **170** is connected to decoder **184** by bus **186**. The decoder **184** provides internal signals to the rest of the microprocessor **50** on lines **188**

Most significant bits **190** of each instruction byte **1-4** location are connected to a 4-input decoder **192** by lines **194**. The output of decoder **192** is connected to memory controller **118** by line **196**. Program counter **130** is connected to memory controller **118** by internal address bus **136**. and the instruction register **108** is connected to the memory controller **118** by the internal data bus **90**. Address/data bus **198** and control bus **200** are connected to the DRAMS **150** (FIG. **3**).

In operation. when the most significant bits **190** of remaining instructions **1–4** are "1" in a clock cycle of the microprocessor **50**. there are no memory reference instructions in the queue. The output of decoder **192** on line **196** requests an instruction fetch ahead by memory controller **118** without interference with other accesses. While the current instructions in instruction register **108** are executing the memory controller **118** obtains the address of the next set of four instructions from program counter **130** and obtains that set of instructions. By the time the current set of instructions has completed execution. the next set of instructions is ready for loading into the instruction register.

Details of the DMA CPU **72** are provided in FIG. **5**. Internal data bus **90** is connected to memory controller **118** and to DMA instruction register **210**. The DMA instruction register **210** is connected to DMA program counter **212** by bus **214**. to transfer size counter **216** by bus **218** and to timed transfer interval counter **220** by bus **222**. The DMA instruction register **210** is also connected to DMA I/O and RAM address register **224** by line **226**. The DMA I/O and RAM address register **224** is connected to the memory controller **118** by memory cycle request line **228** and bus **230**. The DMA program counter **212** is connected to the internal address bus **136** by bus **232**. The transfer size counter **216** is connected to a DMA instruction done decrementer **234** by lines **236** and **238**. The decrementer **234** receives a control input on memory cycle acknowledge line **240**. When transfer size counter **216** has completed its count. it provides a control signal to DMA program counter **212** on line **242**. Timed transfer interval counter **220** is connected to decrementer **244** by lines **246** and **248**. The decrementer **244** receives a control input from a microprocessor system clock on line **250**

The DMA CPU **72** controls itself and has the ability to fetch and execute instructions. It operates as a co-processor to the main CPU **70** (FIG. **2**) for time specific processing.

FIG. **6** shows how the microprocessor **50** is connected to an electrically programmable read only memory (EPROM) **260** by reconfiguring the data lines **52** so that some of the data lines **52** are input lines and some of them are output lines. Data lines **52** D0–D7 provide data to and from corresponding data terminals **262** of the EPROM **260**. Data lines **52** D9–D18 provide addresses to address terminals **264** of the EPROM **260**. Data lines **52** D19–D31 provide inputs from the microprocessor **50** to memory and I/O decode logic **266**. RAS 0/1 control line **268** provides a control signal for determining whether the memory and I/O decode logic provides a DRAM RAS output on line **270** or a column enable output for the EPROM **260** on line **272**. Column address strobe terminal **60** of the microprocessor **50** provides an output enable signal on line **274** to the corresponding terminal **276** of the EPROM **260**.

FIGS. **7** and **8** show the front and back of a one card data processing system **280** incorporating the microprocessor **50**. MSM514258-10 type DRAMs **150** totalling 2 megabytes. a

Motorola 50 MegaHertz crystal oscillator clock **282**, I/O circuits **284** and a 27256 type EPROM **260**. The I/O circuits **284** include a 74HC04 type high speed hex inverter circuit **286**, an IDT39C828 type 10-bit inverting buffer circuit **288**, an IDT39C822 type 10-bit inverting register circuit **290** and two IDT39C823 type 9-bit non-inverting register circuits **292**. The card **280** is completed with a MAX12V type DC—DC converter circuit **294**, 34-pin dual AMP type headers **296**, a coaxial female power connector **298**, and a 3-pin AMP right angle header **300**. The card **280** is a low cost imbeddable product that can be incorporated in larger systems or used as an internal development tool.

The microprocessor **50** is a very high performance (50 MHz) RISC influenced 32-bit CPU designed to work closely with dynamic RAM. Clock for clock the microprocessor **50** approaches the theoretical performance limits possible with a single CPU configuration. Eventually, the microprocessor **50** and any other processor is limited by the bus bandwidth and the number of bus paths. The critical conduit is between the CPU and memory.

One solution to the bus bandwidth/bus path problem is to integrate a CPU directly onto the memory chips, giving every memory a direct bus to the CPU. FIG. **9** shows another microprocessor **310** that is provided integrally with 1 megabit of DRAM **311** in a single integrated circuit **312**. Until the present invention this solution has not been practical, because most high performance CPUs require from 500,000 to 1,000,000 transistors and enormous die sizes just by themselves. The microprocessor **310** is equivalent to the microprocessor **50** in FIGS. **1–8**. The microprocessors **50** and **310** are the most transistor efficient high performance CPUs in existence, requiring fewer than 50,000 transistors for dual processors **70** and **72** (FIG. **2**) or **314** and **316** (less memory). The very high speed of the microprocessors **50** and **310** is to a certain extent a function of the small number of active devices. In essence, the less silicon gets in the way, the faster the electrons can get where they are going.

The microprocessor **310** is therefore the only CPU suitable for integration on the memory chip die **312**. Some simple modifications to the basic microprocessor **50** to take advantage of the proximity to the DRAM array **311** can also increase the microprocessor **50** clock speed by 50 percent, and probably more.

The microprocessor **310** core on board the DRAM die **312** provides most of the speed and functionality required for a large group of applications from automotive to peripheral control. However, the integrated CPU **310**/DRAM **311** concept has the potential to redefine significantly the way multiprocessor solutions can solve a spectrum of very compute intensive problems. The CPU **310**/DRAM **311** combination eliminates the Von Neumann bottleneck by distributing it across numerous CPU/DRAM chips **312**. The microprocessor **310** is a particularly good core for multiprocessing, since it was designed with the SDI targeting array in mind, and provisions were made for efficient interprocessor communications.

Traditional multiprocessor implementations have been very expensive in addition to being unable to exploit fully the available CPU horsepower. Multiprocessor systems have typically been built up from numerous board level or box level computers. The result is usually an immense amount of hardware with corresponding wiring, power consumption and communications problems. By the time the systems are interconnected, as much as 50 percent of the bus speed has been utilized just getting through the interfaces.

In addition, multiprocessor system software has been scarce. A multiprocessor system can easily be crippled by an inadequate load-sharing algorithm in the system software, which allows one CPU to do a great deal of work and the others to be idle. Great strides have been made recently in systems software, and even UNIX V.4 may be enhanced to support multiprocessing. Several commercial products from such manufacturers as DUAL Systems and UNISOFT do a credible job on 68030 type microprocessor systems now.

The microprocessor **310** architecture eliminates most of the interface friction, since up to **64** CPU **310**/RAM **311** processors should be able to intercommunicate without buffers or latches. Each chip **312** has about **40** MIPS raw speed, because placing the DRAM **311** next to the CPU **310** allows the microprocessor **310** instruction cycle to be cut in half, compared to the microprocessor **50**. A **64** chip array of these chips **312** is more powerful than any other existing computer. Such an array fits on a 3×5 card, cost less than a FAX machine, and draw about the same power as a small television.

Dramatic changes in price/performance always reshape existing applications and almost always create new ones. The introduction of microprocessors in the mid 1970s created video games, personal computers, automotive computers, electronically controlled appliances, and low cost computer peripherals.

The integrated circuit **312** will find applications in all of the above areas, plus create some new ones. A common generic parallel processing algorithm handles convolution/Fast Fourier Transform (FFT)/pattern recognition. Interesting product possibilities using the integrated circuit **312** include high speed reading machines, real-time speech recognition, spoken language translation, real-time robot vision, a product to identify people by their faces, and an automotive or aviation collision avoidance system.

A real time processor for enhancing high density television (HDTV) images, or compressing the HDTV information into a smaller bandwidth, would be very feasible. The load sharing in HDTV could be very straightforward. Splitting up the task according to color and frame would require 6, 9 or 12 processors. Practical implementation might require 4 meg RAMs integrated with the microprocessor **310**.

The microprocessor **310** has the following specifications:

CONTROL LINES

4—POWER/GROUND

1—CLOCK

32—DATA I/O

4—SYSTEM CONTROL

EXTERNAL MEMORY FETCH

EXTERNAL MEMORY FETCH AUTOINCREMENT X

EXTERNAL MEMORY FETCH AUTOINCREMENT Y

EXTERNAL MEMORY WRITE

EXTERNAL MEMORY WRITE AUTOINCREMENT X

EXTERNAL MEMORY WRITE AUTOINCREMENT Y

EXTERNAL PROM FETCH

LOAD ALL X REGISTERS

LOAD ALL Y REGISTERS

LOAD ALL PC REGISTERS

EXCHANGE X AND Y

INSTRUCTION FETCH

ADD TO PC

ADD TO X

WRITE MAPPING REGISTER

READ MAPPING REGISTER

REGISTER CONFIGURATION

MICROPROCESSOR **310** CPU **316** CORE
COLUMN LATCH1 (1024 BITS) 32×32 MUX
STACK POINTER (16 BITS)
COLUMN LATCH2 (1024 BITS) 32×32 MUX
RSTACK POINTER (16 BITS)
PROGRAM COUNTER 32 BITS
XO REGISTER 32 BITS (ACTIVATED ONLY FOR ON-CHIP ACCESSES)
YO REGISTER 32 BITS (ACTIVATED ONLY FOR ON-CHIP ACCESSES)
LOOP COUNTER 32 BITS
DMA CPU **314** CORE
DMA PROGRAM COUNTER 24 BITS
INSTRUCTION REGISTER 32 BITS
I/O & RAM ADDRESS REGISTER 32 BITS
TRANSFER SIZE COUNTER 12 BITS
INTERVAL COUNTER 12 BITS

To offer memory expansion for the basic chip **312**, an intelligent DRAM can be produced. This chip will be optimized for high speed operation with the integrated circuit **312** by having three on-chip address registers: Program Counter, X Register and Y register. As a result, to access the intelligent DRAM, no address is required, and a total access cycle could be as short as 10 nsec. Each expansion DRAM would maintain its own copy of the three registers and would be identified by a code specifying its memory address. Incrementing and adding to the three registers will actually take place on the memory chips. A maximum of 64 intelligent DRAM peripherals would allow a large system to be created without sacrificing speed by introducing multiplexers or buffers.

There are certain differences between the microprocessor **310** and the microprocessor **50** that arise from providing the microprocessor **310** on the same die **312** with the DRAM **311**. Integrating the DRAM **311** allows architectural changes in the microprocessor **310** logic to take advantage of existing on-chip DRAM **311** circuitry. Row and column design is inherent in memory architecture. The DRAMs **311** access random bits in a memory array by first selecting a row of 1024 bits, storing them into a column latch, and then selecting one of the bits as the data to be read or written.

The time required to access the data is split between the row access and the column access. Selecting data already stored in a column latch is faster than selecting a random bit by at least a factor of six. The microprocessor **310** takes advantage of this high speed by creating a number of column latches and using them as caches and shift registers. Selecting a new row of information may be thought of as performing a 1024-bit read or write with the resulting immense bus bandwidth.

1. The microprocessor **50** treats its 32-bit instruction register **108** (see FIGS. **2** and **4**) as a cache for four 8-bit instructions. Since the DRAM **311** maintains a 1024-bit latch for the column bits, the microprocessor **310** treats the column latch as a cache for **128** 8-bit instructions. Therefore the next instruction will almost always be already present in the cache. Long loops within the cache are also possible and more useful than the 4 instruction loops in the microprocessor **50**.

2. The microprocessor **50** uses two 16×32-bit deep register arrays **74** and **134** (FIG. **2**) for the parameter stack and the return stack. The microprocessor **310** creates two other 1024-bit column latches to provide the equivalent of two 32×32-bit arrays, which can be accessed twice as fast as a register array.

3. The microprocessor **50** has a DMA capability which can be used for I/O to a video shift register. The microprocessor **310** uses yet another 1024-bit column latch as a long video shift register to drive a CRT display directly. For color displays, three on-chip shift registers could also be used. These shift registers can transfer pixels at a maximum of 100 MHz.

4. The microprocessor **50** accesses memory via an external 32-bit bus. Most of the memory **311** for the microprocessor **310** is on the same die **312**. External access to more memory is made using an 8-bit bus. The result is a smaller die, smaller package and lower power consumption than the microprocessor **50**.

5. The microprocessor **50** consumes about a third of its operating power charging and discharging the I/O pins and associated capacitances. The DRAMs **150** (FIG. **8**) connected to the microprocessor **50** dissipate most of their power in the I/O drivers. A microprocessor **310** system will consume about one-tenth the power of a microprocessor **50** system, since having the DRAM **311** next to the processor **310** eliminates most of the external capacitances to be charged and discharged.

6. Multiprocessing means splitting a computing task between numerous processors in order to speed up the solution. The popularity of multiprocessing is limited by the expense of current individual processors as well as the limited interprocessor communications ability. The microprocessor **310** is an excellent multiprocessor candidate, since the chip **312** is a monolithic computer complete with memory, rendering it low-cost and physically compact.

The shift registers implemented with the microprocessor **310** to perform video output can also be configured as interprocessor communication links. The INMOS transputer attempted a similar strategy, but at much lower speed and without the performance benefits inherent in the microprocessor **310** column latch architecture. Serial I/O is a prerequisite for many multiprocessor topologies because of the many neighbor processors which communicate. A cube has 6 neighbors. Each neighbor communicates using these lines:

DATA IN
CLOCK IN
READY FOR DATA
DATA OUT
DATA READY?
CLOCK OUT

A special start up sequence is used to initialize the on-chip DRAM **311** in each of the processors.

The microprocessor **310** column latch architecture allows neighbor processors to deliver information directly to internal registers or even instruction caches of other chips **312**. This technique is not used with existing processors, because it only improves performance in a tightly coupled DRAM system.

7. The microprocessor **50** architecture offers two types of looping structures: LOOP-IF-DONE and MICRO-LOOP. The former takes an 8-bit to 24-bit operand to describe the entry point to the loop address. The latter performs a loop entirely within the 4 instruction queue and the loop entry point is implied as the first instruction in the queue. Loops entirely within the queue run without external instruction fetches and execute up to three times as fast as the long loop construct. The microprocessor **310** retains both constructs with a few differences. The microprocessor **310** microloop functions in the same fashion as the microprocessor **50** operation, except the queue is 1024-bits or 128 8-bit instructions long. The microprocessor **310** microloop can therefore contain jumps, branches, calls and immediate operations not possible in the 4 8-bit instruction microprocessor **50** queue.

Microloops in the microprocessor **50** can only perform simple block move and compare functions The larger microprocessor **310** queue allows entire digital signal processing or floating point algorithms to loop at high speed in the queue

The microprocessor **50** offers four instructions to redirect execution:

CALL

BRANCH

BRANCH-IF-ZERO

LOOP-IF-NOT-DONE

These instructions take a variable length address operand 8 16 or 24 bits long. The microprocessor **50** next address logic treats the three operands similarly by adding or subtracting them to the current program counter For the microprocessor **310** the 16 and 24-bit operands function in the same manner as the 16 and 24-bit operands in the microprocessor **50** The 8-bit class operands are reserved to operate entirely within the instruction queue. Next address decisions can therefore be made quickly because only 10 bits of addresses are affected rather than 32. There is no carry or borrow generated past the 10 bits.

8 The microprocessor **310** CPU **316** resides on an already crowded DRAM die **312**. To keep chip size as small as possible the DMA processor **72** of the microprocessor **50** has been replaced with a more traditional DMA controller **314** DMA is used with the microprocessor **310** to perform the following functions:

Video output to a CRT

Multiprocessor serial communications

8-bit parallel I/O

The DMA controller **314** can maintain both serial and parallel transfers simultaneously. The following DMA sources and destinations are supported by the microprocessor **310**:

DESCRIPTION	I/O	LINES
1 Video shift register	OUTPUT	1 to 3
2 Multiprocessor serial	BOTH	6 lines/channel
3 8-bit parallel	BOTH	8 data, 4 control

The three sources use separate 1024-bit buffers and separate I/O pins Therefore all three may be active simultaneously without interference.

The microprocessor **310** can be implemented with either a single multiprocessor serial buffer or separate receive and sending buffers for each channel, allowing simultaneous bidirectional communications with six neighbors simultaneously

FIGS **10** and **11** provide details of the PROM DMA used in the microprocessor **50**. The microprocessor **50** executes faster than all but the fastest PROMs PROMS are used in a microprocessor **50** system to store program segments and perhaps entire programs. The microprocessor **50** provides a feature on power-up to allow programs to be loaded from low-cost slow speed PROMs into high speed DRAM for execution The logic which performs this function is part of the DMA memory controller **118**. The operation is similar to DMA, but not identical since four 8-bit bytes must be assembled on the microprocessor **50** chip then written to the DRAM **150**.

The microprocessor **50** directly interfaces to DRAM **150** over a triple multiplexed data and address bus **350**, which carries RAS addresses CAS addresses and data The EPROM **260** on the other hand, is read with non-multiplexed busses. The microprocessor **50** therefore has a special mode which unmultiplexes the data and address lines to read 8 bits of EPROM data Four 8-bit bytes are read in this fashion The multiplexed bus **350** is turned back on, and the data is written to the DRAM **150**

When the microprocessor **50** detects a RESET condition, the processor stops the main CPU **70** and forces a mode 0 (PROM LOAD) instruction into the DMA CPU **72** instruction register. The DMA instruction directs the memory controller to read the EPROM **260** data at 8 times the normal access time for memory Assuming a 50 MHz microprocessor **50**, this means an access time of 320 nsec The instruction also indicates:

The selection address of the EPROM **260** to be loaded.

The number of 32-bit words to transfer

The DRAM **150** address to transfer into

The sequence of activities to transfer one 32-bit word from EPROM **260** to DRAM **150** are:

1. RAS goes low at 352 latching the EPROM **260** select information from the high order address bits. The EPROM **260** is selected
2. Twelve address bits (consisting of what is normally DRAM CAS addresses plus two byte select bits are placed on the bus **350** going to the EPROM **260** address pins These signals will remain on the lines until the data from the EPROM **260** has been read into the microprocessor **50** For the first byte, the byte select bits will be binary 00
3. CAS goes low at 354 enabling the EPROM **260** data onto the lower 8 bits of the external address/data bus **350**. NOTE: It is important to recognize that during this part of the cycle the lower 8 bits of the external data/address bus are functioning as inputs but the rest of the bus is still acting as outputs.
4. The microprocessor **50** latches these eight least significant bits internally and shifts them 8 bits left to shift them to the next significant byte position
5. Steps 2, 3 and 4 are repeated with byte address **01**.
6. Steps 2, 3 and 4 are repeated with byte address **10**.
7. Steps 2 3 and 4 are repeated with byte address **11**
8. CAS goes high at 356, taking the EPROM **260** off the data bus
9. RAS goes high at 358 indicating the end of the EPROM **260** access.
10. RAS goes low at 360 latching the DRAM select information from the high order address bits. At the same time, the RAS address bits are latched into the DRAM **150** The DRAM **150** is selected.
11. CAS goes low at 362, latching the DRAM **150** CAS addresses.
12. The microprocessor **50** places the previously latched EPROM **260** 32-bit data onto the external address/data bus **350**. W goes low at 364 writing the 32 bits into the DRAM **150**
13. W goes high at 366 CAS goes high at 368 The process continues with the next word

FIG. **12** shows details of the microprocessor **50** memory controller **118**. In operation bus requests stay present until they are serviced. CPU **70** requests are prioritized at **370** in the order of: 1 Parameter Stack; 2 Return Stack; 3. Data Fetch; 4 Instruction Fetch. The resulting CPU request signal and a DMA request signal are supplied as bus requests to bus control **372**, which provides a bus grant signal at **374** Internal address bus **136** and a DMA counter **376** provide inputs to a multiplexer **378**. Either a row address or a column address are provided as an output to multiplexed address bus **380** as an output from the multiplexer **378** The multiplexed

address bus 380 and the internal data bus 90 provide address and data inputs, respectively, to multiplexer 382. Shift register 384 supplies row address strobe (RAS) 1 and 2 control signals to multiplexer 386 and column address strobe (CAS) 1 and 2 control signals to multiplexer 388 on lines 390 and 392. The shift register 384 also supplies output enable (OE) and write (W) signals on lines 394 and 396 and a control signal on line 398 to multiplexer 382. The shift register 384 receives a RUN signal on line 400 to generate a memory cycle and supplies a MEMORY READY signal on line 402 when an access is complete.

STACK/REGISTER ARCHITECTURE

Most microprocessors use on-chip registers for temporary storage of variables. The on-chip registers access data faster than off-chip RAM. A few microprocessors use an on-chip push down stack for temporary storage.

A stack has the advantage of faster operation compared to on-chip registers by avoiding the necessity to select source and destination registers. (A math or logic operation always uses the top two stack items as source and the top of stack as destination.) The stack's disadvantage is that it makes some operations clumsy. Some compiler activities in particular require on-chip registers for efficiency.

As shown in FIG. 13, the microprocessor 50 provides both on-chip registers 134 and a stack 74 and reaps the benefits of both.

BENEFITS:

1. Stack math and logic is twice as fast as those available on an equivalent register only machine. Most programmers and optimizing compilers can take advantage of this feature.
2. Sixteen registers are available for on-chip storage of local variables which can transfer to the stack for computation. The accessing of variables is three to four times as fast as available on a strictly stack machine.

The combined stack 74/register 134 architecture has not been used previously due to inadequate understanding by computer designers of optimizing compilers and the mix of transfer versus math/logic instructions.

ADAPTIVE MEMORY CONTROLLER

A microprocessor must be designed to work with small or large memory configurations. As more memory loads are added to the data, address, and control lines, the switching speed of the signals slows down. The microprocessor 50 multiplexes the address/data bus three ways, so timing between the phases is critical. A traditional approach to the problem allocates a wide margin of time between bus phases so that systems will work with small or large numbers of memory chips connected. A speed compromise of as much as 50% is required.

As shown in FIG. 14, the microprocessor 50 uses a feedback technique to allow the processor to adjust memory bus timing to be fast with small loads and slower with large ones. The OUTPUT ENABLE (OE) line 152 from the microprocessor 50 is connected to all memories 150 on the circuit board. The loading on the output enable line 152 to the microprocessor 50 is directly related to the number of memories 150 connected. By monitoring how rapidly OE 152 goes high after a read, the microprocessor 50 is able to determine when the data hold time has been satisfied and place the next address on the bus.

The level of the OE line 152 is monitored by CMOS input buffer 410 which generates an internal READY signal on line 412 to the microprocessor's memory controller. Curves 414 and 416 of the FIG. 15 graph show the difference in rise time likely to be encountered from a lightly to heavily loaded memory system. When the OE line 152 has reached a predetermined level to generate the READY signal, driver 418 generates an OUTPUT ENABLE signal on OE line 152.

SKIP WITHIN THE INSTRUCTION CACHE

The microprocessor 50 fetches four 8-bit instructions each memory cycle and stores them in a 32-bit instruction register 108, as shown in FIG. 16. A class of "test and skip" instructions can very rapidly execute a very fast jump operation within the four instruction cache.

SKIP CONDITIONS:

Always
ACC non-zero
ACC negative
Carry flag equal logic one
Never
ACC equal zero
ACC positive
Carry flag equal logic zero

The SKIP instruction can be located in any of the four byte positions 420 in the 32-bit instruction register 108. If the test is successful, SKIP will jump over the remaining one, two, or three 8-bit instructions in the instruction register 108 and cause the next four-instruction group to be loaded into the register 108. As shown, the SKIP operation is implemented by resetting the 2-bit microinstruction counter 180 to zero on line 422 and simultaneously latching the next instruction group into the register 108. Any instructions following the SKIP in the instruction register are overwritten by the new instructions and not executed.

The advantage of SKIP is that optimizing compilers and smart programmers can often use it in place of the longer conditional JUMP instruction. SKIP also makes possible microloops which exit when the loop counts down or when the SKIP jumps to the next instruction group. The result is very fast code.

Other machines (such as the PDP-8 and Data General NOVA) provide the ability to skip a single instruction. The microprocessor 50 provides the ability to skip up to three instructions.

MICROLOOP IN THE INSTRUCTION CACHE

The microprocessor 50 provides the MICROLOOP instruction to execute repetitively from one to three instructions residing in the instruction register 108. The microloop instruction works in conjunction with the LOOP COUNTER 92 (FIG. 2) connected to the internal data bus 90. To execute a microloop, the program stores a count in LOOP COUNTER 92. MICROLOOP may be placed in the first, second, third, or last byte 420 of the instruction register 108. If placed in the first position, execution will just create a delay equal to the number stored in LOOP COUNTER 92 times the machine cycle. If placed in the second, third, or last byte 420, when the microloop instruction is executed, it will test the LOOP COUNT for zero. If zero, execution will continue with the next instruction. If not zero, the LOOP COUNTER 92 is decremented and the 2-bit microinstruction counter is cleared, causing the preceding instructions in the instruction register to be executed again.

Microloop is useful for block move and search operations. By executing a block move completely out of the instruction register 108, the speed of the move is doubled, since all memory cycles are used by the move rather than being shared with instruction fetching. Such a hardware implementation of microloops is much faster than conventional software implementation of a comparable function.

OPTIMAL CPU CLOCK SCHEME

The designer of a high speed microprocessor must produce a product which operate over wide temperature ranges.

wide voltage swings, and wide variations in semiconductor processing. Temperature, voltage, and process all affect transistor propagation delays. Traditional CPU designs are done so that with the worse case of the three parameters, the circuit will function at the rated clock speed. The result are designs that must be clocked a factor of two slower than their maximum theoretical performance, so they will operate properly in worse case conditions.

The microprocessor **50** uses the technique shown in FIGS. **17–19** to generate the system clock and its required phases. Clock circuit **430** is the familiar "ring oscillator" used to test process performance. The clock is fabricated on the same silicon chip as the rest of the microprocessor **50**.

The ring oscillator frequency is determined by the parameters of temperature, voltage, and process. At room temperature, the frequency will be in the neighborhood of 100 MHZ. At 70 degrees Centigrade, the speed will be 50 MHZ. The ring oscillator **430** is useful as a system clock, with its stages **431** producing phase 0-phase 3 outputs **433** shown in FIG. **19** because its performance tracks the parameters which similarly affect all other transistors on the same silicon die. By deriving system timing from the ring oscillator **430**, CPU **70** will always execute at the maximum frequency possible, but never too fast. For example, if the processing of a particular die is not good resulting in slow transistors, the latches and gates on the microprocessor **50** will operate slower than normal. Since the microprocessor **50** ring oscillator clock **430** is made from the same transistors on the same die as the latches and gates, it too will operate slower (oscillating at a lower frequency), providing compensation which allows the rest of the chip's logic to operate properly.

ASYNCHRONOUS/SYNCHRONOUS CPU

Most microprocessors derive all system timing from a single clock. The disadvantage is that different parts of the system can slow all operations. The microprocessor **50** provides a dual-clock scheme as shown in FIG. **17**, with the CPU **70** operating asynchronously to I/O interface **432** forming part of memory controller **118** (FIG. **2**) and the I/O interface **432** operating synchronously with the external world of memory and I/O devices. The CPU **70** executes at the fastest speed possible using the adaptive ring counter clock **430**. Speed may vary by a factor of four depending upon temperature, voltage, and process. The external world must be synchronized to the microprocessor **50** for operations such as video display updating and disc drive reading and writing. This synchronization is performed by the I/O interface **432**, speed of which is controlled by a conventional crystal clock **434**. The interface **432** processes requests for memory accesses from the microprocessor **50** and acknowledges the presence of I/O data. The microprocessor **50** fetches up to four instructions in a single memory cycle and can perform much useful work before requiring another memory access. By decoupling the variable speed of the CPU **70** from the fixed speed of the I/O interface **432**, optimum performance can be achieved by each. Recoupling between the CPU **70** and the interface **432** is accomplished with handshake signals on lines **436**, with data/addresses passing on bus **90**, **136**.

ASYNCHRONOUS/SYNCHRONOUS CPU IMBEDDED ON A DRAM CHIP

System performance is enhanced even more when the DRAM **311** and CPU **314** (FIG. **9**) are located on the same die. The proximity of the transistors means that DRAM **311** and CPU **314** parameters will closely follow each other. At room temperature, not only would the CPU **314** execute at 100 MHZ, but the DRAM **311** would access fast enough to

keep up. The synchronization performed by the I/O interface **432** would be for DMA and reading and writing I/O ports. In some systems (such as calculators) no I/O synchronization at all would be required, and the I/O clock would be tied to the ring counter clock.

VARIABLE WIDTH OPERANDS

Many microprocessors provide variable width operands. The microprocessor **50** handles operands of 8, 16, or 24 bits using the same op-code. FIG. **20** shows the 32-bit instruction register **108** and the 2-bit microinstruction register **180** which selects the 8-bit instruction. Two classes of microprocessor **50** instructions can be greater than 8-bits, JUMP class and IMMEDIATE. A JUMP or IMMEDIATE op-code is 8-bits, but the operand can be 8, 16 or 24 bits long. This magic is possible because operands must be right justified in the instruction register. This means that the least significant bit of the operand is always located in the least significant bit of the instruction register. The microinstruction counter **180** selects which 8-bit instruction to execute. If a JUMP or IMMEDIATE instruction is decoded, the state of the 2-bit microinstruction counter selects the required 8, 16 or 24 bit operand onto the address or data bus. The unselected 8-bit bytes are loaded with zeros by operation of decoder **440** and gates **442**. The advantage of this technique is the saving of a number of op-codes required to specify the different operand sizes in other microprocessors.

TRIPLE STACK CACHE

Computer performance is directly related to the system memory bandwidth. The faster the memories, the faster the computer. Fast memories are expensive, so techniques have been developed to move a small amount of high-speed memory around to the memory addresses where it is needed. A large amount of slow memory is constantly updated by the fast memory, giving the appearance of a large fast memory array. A common implementation of the technique is known as a high-speed memory cache. The cache may be thought of as fast acting shock absorber smoothing out the bumps in memory access. When more memory is required than the shock can absorb, it bottoms out and slow speed memory is accessed. Most memory operations can be handled by the shock absorber itself.

The microprocessor **50** architecture has the ALU **80** (FIG. **2**) directly coupled to the top two stack locations **76** and **78**. The access time of the stack **74** therefore directly affects the execution speed of the processor. The microprocessor **50** stack architecture is particularly suitable to a triple cache technique, shown in FIG. **21** which offers the appearance of a large stack memory operating at the speed of on-chip latches **450**. Latches **450** are the fastest form of memory device built on the chip, delivering data in as little as 3 nsec. However latches **450** require large numbers of transistors to construct. On-chip RAM **452** requires fewer transistors than latches, but is slower by a factor of five (15 nsec access). Off-chip RAM **150** is the slowest storage of all. The microprocessor **50** organizes the stack memory hierarchy as three interconnected stacks **450**, **452** and **454**. The latch stack **450** is the fastest and most frequently used. The on-chip RAM stack **452** is next. The off-chip RAM stack **454** is slowest. The stack modulation determines the effective access time of the stack. If a group of stack operations never push or pull more than four consecutive items on the stack, operations will be entirely performed in the 3 nsec latch stack. When the four latches **456** are filled, the data in the bottom of the latch stack **450** is written to the top of the on-chip RAM stack **452**. When the sixteen locations **458** in the on-chip RAM stack **452** are filled, the data in the bottom of the on-chip RAM stack **452** is written to the top of the off-chip

RAM stack 454 When popping data off a full stack 450, four pops will be performed before stack empty line 460 from the latch stack pointer 462 transfers data from the on-chip RAM stack 452. By waiting for the latch stack 450 to empty before performing the slower on-chip RAM access, the high effective speed of the latches 456 are made available to the processor. The same approach is employed with the on-chip RAM stack 452 and the off-chip RAM stack 454.

POLYNOMIAL GENERATION INSTRUCTION

Polynomials are useful for error correction, encryption data compression and fractal generation. A polynomial is generated by a sequence of shift and exclusive OR operations. Special chips are provided for this purpose in the prior art.

The microprocessor **50** is able to generate polynomials at high speed without external hardware by slightly modifying how the ALU **80** works. As shown in FIG. **22**, a polynomial is generated by loading the "order" (also known as the feedback terms) into C Register **470**. The value thirty one (resulting in 32 iterations) is loaded into DOWN COUNTER **472**. A register **474** is loaded with zero. B register **476** is loaded with the starting polynomial value. When the POLY instruction executes, C register **470** is exclusively ORed with A register **474** if the least significant bit of B register **476** is a one. Otherwise, the contents of the A register **474** passes through the ALU **80** unaltered. The combination of A and B is then shifted right (divided by 2) with shifters **478** and **480**. The operation automatically repeats the specified number of iterations, and the resulting polynomial is left in A register **474**.

FAST MULTIPLY

Most microprocessors offer a 16×16 or 32×32 bit multiply instruction. Multiply when performed sequentially takes one shift/add per bit, or 32 cycles for 32 bit data. The microprocessor **50** provides a high speed multiply which allows multiplication by small numbers using only a small number of cycles. FIG. **23** shows the logic used to implement the high speed algorithm. To perform a multiply, the size of the multiplier less one is placed in the DOWN COUNTER **472**. For a four bit multiplier, the number three would be stored in the DOWN COUNTER **472**. Zero is loaded into the A register **474**. The multiplier is written bit reversed into the B Register **476**. For example, a bit reversed five (binary 0101) would be written into B as 1010. The multiplicand is written into the C register **470**. Executing the FAST MULT instruction will leave the result in the A Register **474** when the count has been completed. The fast multiply instruction is important because many applications scale one number by a much smaller number. The difference in speed between multiplying a 32×32 bit and a 32×4 bit is a factor of 8. If the least significant bit of the multiplier is a "ONE", the contents of the A register **474** and the C register **470** are added. If the least significant bit of the multiplier is a "ZERO", the contents of the A register are passed through the ALU **80** unaltered. The output of the ALU **80** is shifted left by shifter **482** in each iteration. The contents of the B register **476** are shifted right by the shifter **480** in each iteration.

INSTRUCTION EXECUTION PHILOSOPHY

The microprocessor **50** uses high speed D latches in most of the speed critical areas. Slower on-chip RAM is used as secondary storage.

The microprocessor **50** philosophy of instruction execution is to create a hierarchy of speed as follows:

Logic and D latch transfers	1 cycle	20 nsec
Math	2 cycles	40 nsec
Fetch/store on-chip RAM	2 cycles	40 nsec
Fetch/store in current RAS page	4 cycles	80 nsec
Fetch/store with RAS cycle	11 cycles	220 nsec

With a 50 MHZ clock, many operations can be performed in 20 nsec. and almost everything else in 40 nsec.

To maximize speed, certain techniques in processor design have been used. They include:

Eliminating arithmetic operations on addresses

Fetching up to four instructions per memory cycle

Pipelineless instruction decoding

Generating results before they are needed.

Use of three level stack caching

PIPELINE PHILOSOPHY

Computer instructions are usually broken down into sequential pieces, for example: fetch, decode, register read, execute, and store. Each piece will require a single machine cycle. In most Reduced Instruction Set Computer (RISC) chips, instruction require from three to six cycles.

RISC instructions are very parallel. For example, each of 70 different instructions in the SPARC (SUN Computer's RISC chip) has five cycles. Using a technique called "pipelining", the different phases of consecutive instructions can be overlapped.

To understand pipelining, think of building five residential homes. Each home will require, in sequence, a foundation, framing, plumbing and wiring, roofing, and interior finish. Assume that each activity takes one week. To build one house will take five weeks.

But what if you want to build an entire subdivision? You have only one of each work crew, but when the foundation men finish on the first house, you immediately start them on the second one, and so on. At the end of five weeks, the first home is complete, but you also have five foundations. If you have kept the framing, plumbing, roofing, and interior guys all busy, from five weeks on, a new house will be completed each week.

This is the way a RISC chip like SPARC appears to execute an instruction in a single machine cycle. In reality, a RISC chip is executing one fifth of five instructions each machine cycle. And if five instructions stay in sequence, an instruction will be completed each machine cycle.

The problems with a pipeline are keeping the pipe full with instructions. Each time an out of sequence instruction such as a BRANCH or CALL occurs, the pipe must be refilled with the next sequence. The resulting dead time to refill the pipeline can become substantial when many IF/THEN/ELSE statements or subroutines are encountered.

THE PIPELINE APPROACH

The microprocessor **50** has no pipeline as such. The approach of this microprocessor to speed is to overlap instruction fetching with execution of the previously fetched instruction(s). Beyond that, over half the instructions (the most common ones) execute entirely in a single machine cycle of 20 nsec. This is possible because:

1. Instruction decoding resolves in 2.5 nsec.
2. Incremented/decremented and some math values are calculated before they are needed, requiring only a latching signal to execute.
3. Slower memory is hidden from high speed operations by high-speed D latches which access in 4 nsec.

The disadvantage for this microprocessor is a more complex chip design process. The advantage for the chip user is faster

ultimate throughput since pipeline stalls cannot exist. Pipeline synchronization with availability flag bits and other such pipeline handling is not required by this microprocessor.

For example, in some RISC machines an instruction which tests a status flag may have to wait for up to four cycles for the flag set by the previous instruction to be available to be tested. Hardware and software debugging is also somewhat easier because the user doesn't have to visualize five instructions simultaneously in the pipe.

OVERLAPPING INSTRUCTION FETCH/EXECUTE

The slowest procedure the microprocessor **50** performs is to access memory. Memory is accessed when data is read or written. Memory is also read when instructions are fetched. The microprocessor **50** is able to hide fetch of the next instruction behind the execution of the previously fetched instruction(s). The microprocessor **50** fetches instructions in 4-byte instruction groups. An instruction group may contain from one to four instructions. The amount of time required to execute the instruction group ranges from 4 cycles for simple instructions to 64 cycles for a multiply.

When a new instruction group is fetched, the microprocessor instruction decoder looks at the most significant bit of all four of the bytes. The most significant bit of an instruction determines if a memory access is required. For example, CALL, FETCH, and STORE all require a memory access to execute. If all four bytes have nonzero most significant bits, the microprocessor initiates the memory fetch of the next sequential 4-byte instruction group. When the last instruction in the group finishes executing, the next 4-byte instruction group is ready and waiting on the data bus needing only to be latched into the instruction register. If the 4-byte instruction group required four or more cycles to execute and the next sequential access was a column address strobe (CAS) cycle, the instruction fetch was completely overlapped with execution.

INTERNAL ARCHITECTURE

The microprocessor **50** architecture consists of the following:

PARAMETER STACK <-->	ALU* <-->	Y REGISTER RETURN STACK
<--32 BITS--> 16 DEEP		<--32 BITS--> 16 DEEP
Used for math and logic		Used for subroutine and interrupt return addresses as well as local variables
Push down stack. Can overflow into off-chip RAM.		Push down stack. Can overflow into off-chip RAM. Can also be accessed relative to top of stack
LOOP COUNTER		(32-bits, can decrement by 1) Used by class of test and loop instructions.
X REGISTER		(32-bits, can increment or decrement by 4). Used to point to RAM locations
PROGRAM COUNTER		(32-bits, increments by 4). Points to 4-byte instruction groups in RAM
INSTRUCTION REG		(32-Bits). Holds 4-byte instruction groups while they are being decoded and executed

* Math and logic operations use the TOP item and NEXT to top Parameter Stack items as the operands. The result is pushed onto the Parameter Stack.

* Return addresses from subroutines are placed on the Return Stack. The Y REGISTER is used as a pointer to RAM locations. Since the Y REGISTER is the top item of the Return Stack, nesting of indices is straightforward.

MODE—A register with mode and status bits

MODE-BITS:

Slow down memory accesses by 8 if "1". Run full speed if "0". (Provided for access to slow EPROM.)

Divide the system clock by 1023 if "1" to reduce power consumption. Run full speed if "0". (On-chip counters slow down if this bit is set.)

Enable external interrupt **1**.

Enable external interrupt **2**.

Enable external interrupt **3**

Enable external interrupt **4**

Enable external interrupt **5**

Enable external interrupt **6**

Enable external interrupt **7**.

ON-CHIP MEMORY LOCATIONS:

MODE-BITS

DMA-POINTER

DMA-COUNTER

STACK-POINTER—Pointer into Parameter Stack

STACK-DEPTH—Depth of on-chip Parameter Stack

RSTACK-POINTER—Pointer into Return Stack

RSTACK-DEPTH—Depth of on-chip Return Stack

ADDRESSING MODE HIGH POINTS

The data bus is 32-bits wide. All memory fetches and stores are 32-bits. Memory bus addresses are 30 bits. The least significant 2 bits are used to select one-of-four bytes in some addressing modes. The Program Counter, X Register and Y Register are implemented as D latches with their outputs going to the memory address bus and the bus incrementer/decrementer. Incrementing one of these registers can happen quickly, because the incremented value has already rippled through the inc/dec logic and need only be clocked into the latch. Branches and Calls are made to 32-bit word boundaries.

INSTRUCTION SET

32-BIT INSTRUCTION FORMAT

The thirty two bit instructions are CALL, BRANCH, BRANCH-IF-ZERO, and LOOP-IF-NOT-DONE. These instructions require the calculation of an effective address. In many computers, the effective address is calculated by adding or subtracting an operand with the current Program Counter. This math operation requires from four to seven machine cycles to perform and can definitely bog down machine execution. The microprocessor's strategy is to perform the required math operation at assembly or linking time and do a much simpler "Increment to next page" or "Decrement to previous page" operation at run time. As a result, the microprocessor branches execute in a single cycle.

24-BIT OPERAND FORM:

Byte **1** Byte **2** Byte **3** Byte **4**

WWWWWW XX—YYYYYYYY—YYYYYYYY—YYYYYYYY

With a 24-bit operand, the current page is considered to be defined by the most significant 6 bits of the Program Counter.

16-BIT OPERAND FORM:

QQQQQQQQ—WWWWWW XX—YYYYYYYY—YYYYYYYY

With a 16-bit operand, the current page is considered to be defined by the most significant 14 bits of the Program Counter.

8-BIT OPERAND FORM:

QQQQQQQQ—QQQQQQQQ—WWWWWW XX—YYYYYYYY

With an 8-bit operand, the current page is considered to be defined by the most significant 22 bits of the Program Counter.

QQQQQQQQ—Any 8-bit instruction

WWWWWW—Instruction op-code.

XX—Select how the address bits will be used:

00—Make all high-order bits zero (Page zero addressing)

01—Increment the high-order bits. (Use next page)

10—Decrement the high-order bits (Use previous page)

11—Leave the high-order bits unchanged. (Use current page)

YYYYYYYY—The address operand field. This field is always shifted left two bits (to generate a word rather than byte address) and loaded into the Program Counter. The microprocessor instruction decoder figures out the width of the operand field by the location of the instruction op-code in the four bytes

The compiler or assembler will normally use the shortest operand required to reach the desired address so that the leading bytes can be used to hold other instructions. The effective address is calculated by combining:

The current Program Counter.

The 8, 16, or 24 bit address operand in the instruction. Using one of the four allowed addressing modes.

EXAMPLES OF EFFECTIVE ADDRESS CALCULATION

EXAMPLE 1 :

Byte 1	Byte 2	Byte 3	Byte 4
QQQQQQQQ	QQQQQQQQ	00000011	10011000

The "QQQQQQQQs" in Byte 1 and 2 indicate space in the 4-byte memory fetch which could be hold two other instructions to be executed prior to the CALL instruction. Byte 3 indicates a CALL instruction (six zeros) in the current page (indicated by the 11 bits) Byte 4 indicates that the hexadecimal number 98 will be forced into the Program Counter bits **2** through **10**. (Remember, a CALL or BRANCH always goes to a word boundary so the two least significant bits are always set to zero). The effect of this instruction would be to CALL a subroutine at WORD location HEX 98 in the current page. The most significant 22 bits of the Program Counter define the current page and will be unchanged.

EXAMPLE 2 :

Byte 1	Byte 2	Byte 3	Byte 4
000001 01	00000001	00000000	00000000

If we assume that the Program Counter was HEX 0000 0156 which is binary:

00000000 00000000 00000001 01010110=OLD PROGRAM COUNTER

Byte **1** indicates a BRANCH instruction op code (000001) and "01" indicates select the next page. Byte **2**, **3**, and **4** are the address operand. These 24-bits will be shifted to the left two places to define a WORD address. HEX 0156 shifted left two places is HEX 0558. Since this is a 24-bit operand instruction, the most significant 6 bits of the Program Counter define the current page. These six bits will be incremented to select the next page. Executing this instruction will cause the Program Counter to be loaded with HEX 0400 0558 which is binary:

00000100 00000000 00000101 01011000 = NEW PROGRAM COUNTER.

INSTRUCTIONS

CALL-LONG

0000 00XXX – YYYYYYYY – YYYYYYYY – YYYYYYYY

Load the Program Counter with the effective WORD address specified. Push the current PC contents onto the RETURN STACK.

OTHER EFFECTS: CARRY or modes, no effect. May cause Return Stack to force an external memory cycle if on-chip Return Stack is full

BRANCH

0000 01XX—YYYYYYYY—YYYYYYYY—YYYYYYYY

Load the Program Counter with the effective WORD address specified

OTHER EFFECTS: NONE

BRANCH-IF-ZERO

0000 10XX—YYYYYYYY—YYYYYYYY—YYYYYYYY

Test the TOP value on the Parameter Stack. If the value is equal to zero, load the Program Counter with the effective WORD address specified. If the TOP value is not equal to zero, increment the Program Counter and fetch and execute the next instruction.

OTHER EFFECTS: NONE

LOOP-IF-NOT-DONE

0000 11YY—(XXXX XXXX)—(XXXX XXXX)—(XXXX XXXX)

If the LOOP COUNTER is not zero, load the Program Counter with the effective WORD address specified. If the LOOP COUNTER is zero, decrement the LOOP COUNTER, increment the Program Counter and fetch and execute the next instruction.

OTHER EFFECTS: NONE

8-BIT INSTRUCTIONS PHILOSOPHY

Most of the work in the microprocessor **50** is done by the 8-bit instructions. Eight bit instructions are possible with the microprocessor because of the extensive use of implied stack addressing. Many 32-bit architectures use 8-bits to specify the operation to perform but use an additional 24-bits to specify two sources and a destination.

For math and logic operations, the microprocessor **50** exploits the inherent advantage of a stack by designating the source operand(s) as the top stack item and the next stack item. The math or logic operation is performed, the operands are popped from the stack, and the result is pushed back on the stack. The result is a very efficient utilization of instruction bits as well as registers. A comparable situation exists between Hewlett Packard calculators (which use a stack) and Texas Instrument calculators which don't. The identical operation on an HP will require one half to one third the keystrokes of the TI.

The availability of 8-bit instructions also allows another architectural innovation, the fetching of four instructions in a single 32-bit memory cycle. The advantages of fetching multiple instructions are:

Increased execution speed even with slow memories.

Similar performance to the Harvard (separate data and instruction busses) without the expense

Opportunities to optimize groups of instructions.

The capability to perform loops within this mini-cache.

The microloops inside the four instruction group are effective for searches and block moves

SKIP INSTRUCTIONS

The microprocessor **50** fetches instructions in 32-bit chunks called 4-byte instruction groups These four bytes may contain four 8-bit instructions or some mix of 8-bit and 16 or 24-bit instructions. SKIP instructions in the microprocessor skip any remaining instructions in a 4-byte instruction group and cause a memory fetch to get the next 4-byte instruction group. Conditional SKIPs when combined with 3-byte BRANCHES will create conditional BRANCHES. SKIPs may also be used in situations when no use can be made of the remaining bytes in a 4-instruction group A SKIP executes in a single cycle. whereas a group of three NOPs would take three cycles.

SKIP-ALWAYS—Skip any remaining instructions in this 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group.

SKIP-IF-ZERO—If the TOP item of the Parameter Stack is zero. skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the TOP item is not zero. execute the next sequential instruction.

SKIP-IF-POSITIVE—If the TOP item of the Parameter Stack has a the most significant bit (the sign bit) equal to "0" skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group If the TOP item is not "0" execute the next sequential instruction.

SKIP-IF-NO-CARRY—If the CARRY flag from a SHIFT or arithmetic operation is not equal to "1". skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the CARRY is equal to "1". execute the next sequential instruction.

SKIP-NEVER Execute the next sequential (NOP) instruction. (Delay one machine cycle).

SKIP-IF-NOT-ZERO—If the TOP item on the Parameter Stack is not equal to "0". skip any remaining instructions in the 4-byte instruction group Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the TOP item is equal "0". execute the next sequential instruction.

SKIP-IF-NEGATIVE—If the TOP item on the Parameter Stack has its most significant bit (sign bit) set to "1". skip any remaining instructions in the 4-byte instruction group Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the TOP item has its most significant bit set to "0" execute the next sequential instruction.

SKIP-IF-CARRY—If the CARRY flag is set to "1" as a result of SHIFT or arithmetic operation. skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group If the CARRY flag is "0". execute the next sequential instruction

MICROLOOPS

Microloops are a unique feature of the microprocessor architecture which allows controlled looping within a 4-byte instruction group. A microloop instruction tests the LOOP COUNTER for "0" and may perform an additional test. If the LOOP COUNTER is not "0" and the test is met. instruction execution continues with the first instruction in the 4-byte instruction group. and the LOOP COUNTER is decremented A microloop instruction will usually be the last byte in a 4-byte instruction group. but it can be any byte If the LOOP COUNTER is "0" or the test is not met. instruction execution continues with the next instruction. If the microloop is the last byte in the 4-byte instruction group. the most significant 30-bits of the Program Counter are incremented and the next 4-byte instruction group is fetched from memory On a termination of the loop on LOOP COUNTER equal to "0". the LOOP COUNTER will remain at "0". Microloops allow short iterative work such as moves and searches to be performed without slowing down to fetch instructions from memory.

EXAMPLE:

Byte 1 FETCH-VIA-X-AUTOINCREMENT	Byte 2 STORE-VIA-Y-AUTOINCREMENT
Byte 3 ULOOP-UNTIL-DONE	Byte 4 QQQQQQQQ

This example will perform a block move. To initiate the transfer. X will be loaded with the starting address of the source. Y will be loaded with the starting address of the destination. The LOOP COUNTER will be loaded with the number of 32-bit words to move The microloop will FETCH and STORE and count down the LOOP COUNTER until it reaches zero. QQQQQQQQ indicates any instruction can follow.

MICROLOOP INSTRUCTIONS

ULOOP-UNTIL-DONE—If the LOOP COUNTER is not "0". continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER If the LOOP COUNTER is "0". continue execution with the next instruction.

ULOOP-IF-ZERO—If the LOOP COUNTER is not "0" and the TOP item on the Parameter Stack is "0". continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the TOP item is "1". continue execution with the next instruction

ULOOP-IF-POSITIVE—If the LOOP COUNTER is not "0" and the most significant bit (sign bit) is "0" continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the TOP item is "1". continue execution with the next instruction.

ULOOP-IF-NOT-CARRY-CLEAR—If the LOOP COUNTER is not "0" and the floating point exponents found in TOP and NEXT are not aligned. continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the exponents are aligned. continue execution with the next instruction. This instruction is specifically designed for combination with special SHIFT instructions to align two floating point numbers.

ULOOP-NEVER—(DECREMENT-LOOP-COUNTER) Decrement the LOOP COUNTER Continue execution with the next instruction.

ULOOP-IF-NOT-ZERO—If the LOOP COUNTER is not "0" and the TOP item of the Parameter Stack is "0", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the TOP item is "1", continue execution with the next instruction.

ULOOP-IF-NEGATIVE—If the LOOP COUNTER is not "0" and the most significant bit (sign bit) of the TOP item of the Parameter Stack is "1", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the most significant bit of the Parameter Stack is "0", continue execution with the next instruction

ULOOP-IF-CARRY-SET—If the LOOP COUNTER is not "0" and the exponents of the floating point numbers found in TOP and NEXT are not aligned, continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the exponents are aligned, continue execution with the next instruction.

RETURN FROM SUBROUTINE OR INTERRUPT

Subroutine calls and interrupt acknowledgements cause a redirection of normal program execution. In both cases, the current Program Counter is pushed onto the Return Stack so the microprocessor can return to its place in the program after executing the subroutine or interrupt service routine.

NOTE: When a CALL to subroutine or interrupt is acknowledged the Program Counter has already been incremented and is pointing to the 4-byte instruction group following the 4-byte group currently being executed. The instruction decoding logic allows the microprocessor to perform a test and execute a return conditional on the outcome of the test in a single cycle. A RETURN pops an address from the Return Stack and stores it to the Program Counter.

RETURN INSTRUCTIONS

RETURN-ALWAYS—Pop the top item from the Return Stack and transfer it to the Program Counter.

RETURN-IF-ZERO—If the TOP item on the Parameter Stack is "0", pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

RETURN-IF-POSITIVE—If the most significant bit (sign bit) of the TOP item on the Parameter Stack is a "0", pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction

RETURN-IF-CARRY-CLEAR—If the exponents of the floating point numbers found in TOP and NEXT are not aligned, pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

RETURN-NEVER—Execute the next instruction (NOP)

RETURN-IF-NOT-ZERO—If the TOP item on the Parameter Stack is not "0", pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

RETURN-IF-NEGATIVE—If the most significant bit (sign bit) of the TOP item on the Parameter Stack is a "1", pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

RETURN-IF-CARRY-SET—If the exponents of the floating point numbers found in TOP and NEXT are aligned, pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.

HANDLING MEMORY FROM DYNAMIC RAM

The microprocessor **50**, like any RISC type architecture, is optimized to handle as many operations as possible

on-chip for maximum speed. External memory operations take from 80 nsec. to 220 nsec. compared with on-chip memory speeds of from 4 nsec. to 30 nsec. There are times when external memory must be accessed.

External memory is accessed using three registers:

X-REGISTER—A 30-bit memory pointer which can be used for memory access and simultaneously incremented or decremented.

Y-REGISTER—A 30-bit memory pointer which can be used for memory access and simultaneously incremented or decremented

PROGRAM-COUNTER—A 30-bit memory pointer normally used to point to 4-byte instruction groups. External memory may be accessed at addresses relative to the PC. The operands are sometimes called "Immediate" or "Literal" in other computers. When used as memory pointer, the PC is also incremented after each operation.

MEMORY LOAD & STORE INSTRUCTIONS

FETCH-VIA-X—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack. X is unchanged.

FETCH-VIA-Y—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack. Y is unchanged.

FETCH-VIA-X-AUTOINCREMENT—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack. After fetching, increment the most significant 30 bits of X to point to the next 32-bit word address.

FETCH-VIA-Y-AUTOINCREMENT—Fetch the 32-bit memory content pointed to by Y and push it onto the Parameter Stack. After fetching, increment the most significant 30 bits of Y to point to the next 32-bit word address.

FETCH-VIA-X-AUTODECREMENT—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack. After fetching decrement the most significant 30 bits of X to point to the previous 32-bit word address.

FETCH-VIA-Y-AUTODECREMENT—Fetch the 32-bit memory content pointed to by Y and push it onto the Parameter Stack. After fetching, decrement the most significant 30 bits of Y to point to the previous 32-bit word address

STORE-VIA-X—Pop the top item of the Parameter Stack and store it in the memory location pointed to by X. X is unchanged.

STORE-VIA-Y—Pop the top item of the Parameter Stack and store it in the memory location pointed to by Y. Y is unchanged.

STORE-VIA-X-AUTOINCREMENT—Pop the top item of the Parameter Stack and store it in the memory location pointed to by X. After storing increment the most significant 30 bits of X to point to the next 32-bit word address.

STORE-VIA-Y-AUTOINCREMENT—Pop the top item of the Parameter Stack and store it in the memory location pointed to by Y. After storing, increment the most significant 30 bits of Y to point to the next 32-bit word address.

STORE-VIA-X-AUTODECREMENT—Pop the top item of the Parameter Stack and store it in the memory location pointed to by X. After storing decrement the most significant 30 bits of X to point to the previous 32-bit word address.

STORE-VIA-Y-AUTODECREMENT—Pop the top item of the Parameter Stack and store it in the memory location pointed to by Y. After storing, decrement the most significant 30 bits of Y to point to the previous 32-bit word address

FETCH-VIA-PC—Fetch the 32-bit memory content pointed to by the Program Counter and push it onto the Parameter

Stack. After fetching, increment the most significant 30 bits of the Program Counter to point to the next 32-bit word address.

*NOTE When this instruction executes, the PC is pointing to the memory location following the instruction. The effect is of loading a 32-bit immediate operand. This is an 8-bit instruction and therefore will be combined with other 8-bit instructions in a 4-byte instruction fetch. It is possible to have from one to four FETCH-VIA-PC instructions in a 4-byte instruction fetch. The PC increments after each execution of FETCH-VIA-PC. so it is possible to push four immediate operands on the stack. The four operands would be the found in the four memory locations following the instruction.

BYTE-FETCH-VIA-X—Fetch the 32-bit memory content pointed to by the most significant 30 bits of X. Using the two least significant bits of X. select one of four bytes from the 32-bit memory fetch right justify the byte in a 32-bit field and push the selected byte preceded by leading zeros onto the Parameter Stack.

BYTE-STORE-VIA-X—Fetch the 32-bit memory content pointed to by the most significant 30 bits of X. Pop the TOP item from the Parameter Stack.

Using the two least significant bits of X place the least significant byte into the 32-bit memory data and write the 32-bit entity back to the location pointed to by the most significant 30 bits of X.

OTHER EFFECTS OF MEMORY ACCESS INSTRUCTIONS:

Any FETCH instruction will push a value on the Parameter Stack **74**. If the on-chip stack is full. the stack will overflow into off-chip memory stack resulting in an additional memory cycle. Any STORE instruction will pop a value from the Parameter Stack **74**. If the on-chip stack is empty. a memory cycle will be generated to fetch a value from off-chip memory stack.

HANDLING ON-CHIP VARIABLES

High-level languages often allow the creation of LOCAL VARIABLES. These variables are used by a particular procedure and discarded. In cases of nested procedures layers of these variables must be maintained. On-chip storage is up to five times faster than off-chip RAM. so a means of keeping local variables on-chip can make operations run faster. The microprocessor **50** provides the capability for both on-chip storage of local variables and nesting of multiple levels of variables through the Return Stack.

The Return Stack **134** is implemented as 16 on-chip RAM locations. The most common use for the Return Stack **134** is storage of return addresses from subroutines and interrupt calls. The microprocessor allows these 16 locations to also be used as addressable registers. The 16 locations may be read and written by two instructions which indicate a Return Stack relative address from 0–15. When high-level procedures are nested. the current procedure variables push the previous procedure variables further down the Return Stack **134**. Eventually. the Return Stack will automatically overflow into off-chip RAM.

ON-CHIP VARIABLE INSTRUCTIONS

READ-LOCAL-VARIABLE XXXX—Read the XXXXth location relative to the top of the Return Stack. (XXXX is a binary number from 0000–1111). Push the item read onto the Parameter Stack.

OTHER EFFECTS: If the Parameter Stack is full. the push operation will cause a memory cycle to be generated as one item of the stack is automatically stored to external RAM. The logic which selects the location performs a modulo 16 subtraction. If four local variables have been pushed onto the Return Stack. and an instruction attempts to READ the fifth item. unknown data will be returned.

WRITE-LOCAL-VARIABLE XXXX—Pop the TOP item of the Parameter Stack and write it into the XXXXth location relative to the top of the Return Stack. (XXXX is a binary number from 0000–1111.)

OTHER EFFECTS: If the Parameter Stack is empty. the pop operation will cause a memory cycle to be generated to fetch the Parameter Stack item from external RAM. The logic which selects the location performs a modulo 16 subtraction. If four local variables have been pushed onto the Return Stack. and an instruction attempts to WRITE to the fifth item. it is possible to clobber return addresses or wreak other havoc.

REGISTER AND FLIP-FLOP TRANSFER AND PUSH INSTRUCTIONS

DROP—Pop the TOP item from the Parameter Stack and discard it.

SWAP—Exchange the data in the TOP Parameter Stack location with the data in the NEXT Parameter Stack location.

DUP—Duplicate the TOP item on the Parameter Stack and push it onto the Parameter Stack.

PUSH-LOOP-COUNTER—Push the value in LOOP COUNTER onto the Parameter Stack.

POP-RSTACK-PUSH-TO-STACK—Pop the top item from the Return Stack and push it onto the Parameter Stack.

PUSH-X-REG—Push the value in the X Register onto the Parameter Stack.

PUSH-STACK-POINTER—Push the value of the Parameter Stack pointer onto the Parameter Stack.

PUSH-RSTACK-POINTER—Push the value of the Return Stack pointer onto the Return Stack.

PUSH-MODE-BITS—Push the value of the MODE REGISTER onto the Parameter Stack.

PUSH-INPUT—Read the 10 dedicated input bits and push the value (right justified and padded with leading zeros) onto the Parameter Stack.

SET-LOOP-COUNTER—Pop the TOP value from the Parameter Stack and store it into LOOP COUNTER.

POP-STACK-PUSH-TO-RSTACK—Pop the TOP item from the Parameter Stack and push it onto the Return Stack.

SET-X-REG—Pop the TOP item from the Parameter Stack and store it into the X Register.

SET-STACK-POINTER—Pop the TOP item from the Parameter Stack and store it into the Stack Pointer.

SET-RSTACK-POINTER—Pop the TOP item from the Parameter Stack and store it into the Return Stack Pointer.

SET-MODE-BITS—Pop the TOP value from the Parameter Stack and store it into the MODE BITS.

SET-OUTPUT—Pop the TOP item from the Parameter Stack and output it to the 10 dedicated output bits.

OTHER EFFECTS: Instructions which push or pop the Parameter Stack or Return Stack may cause a memory cycle as the stacks overflow back and forth between on-chip and off-chip memory.

LOADING A SHORT LITERAL

A special case of register transfer instruction is used to push an 8-bit literal onto the Parameter Stack. This instruction requires that the 8-bits to be pushed reside in the last byte of a 4-byte instruction group. The instruction op-code loading the literal may reside in ANY of the other three bytes in the instruction group.

EXAMPLE:

BYTE 1 LOAD-SHORT-LITERAL	BYTE 2 QQQQQQQQ	BYTE 3 QQQQQQQQ
BYTE 4 00001111		

In this example QQQQQQQQ indicates any other 8-bit instruction. When Byte 1 is executed, binary 00001111 (HEX 0f) from Byte 4 will be pushed (right justified and padded by leading zeros) onto the Parameter Stack. Then the instructions in Byte 2 and Byte 3 will execute. The microprocessor instruction decoder knows not to execute Byte 4. It is possible to push three identical 8-bit values as follows:

BYTE 1 LOAD-SHORT-LITERAL	BYTE 2 LOAD-SHORT-LITERAL
BYTE 3 LOAD-SHORT-LITERAL	BYTE 4 00001111

SHORT-LITERAL-INSTRUCTION

LOAD-SHORT-LITERAL—Push the 8-bit value found in Byte 4 of the current 4-byte instruction group onto the Parameter Stack.

LOGIC INSTRUCTIONS

Logical and math operations use the stack for the source of one or two operands and as the destination for results. The stack organization is a particularly convenient arrangement for evaluating expressions. TOP indicates the top value on the Parameter Stack 74. NEXT indicates the next to top value on the Parameter Stack 74.

AND—Pop TOP and NEXT from the Parameter Stack, perform the logical AND operation on these two operands, and push the result onto the Parameter Stack.

OR—Pop TOP and NEXT from the Parameter Stack, perform the logical OR operation on these two operands, and push the result onto the Parameter Stack.

XOR—Pop TOP and NEXT from the Parameter Stack, perform the logical exclusive OR on these two operands, and push the result onto the Parameter Stack.

BIT-CLEAR—Pop TOP and NEXT from the Parameter Stack, toggle all bits in NEXT, perform the logical AND operation on TOP, and push the result onto the Parameter Stack. (Another way of understanding this instruction is thinking of it as clearing all bits in TOP that are set in NEXT.)

MATH INSTRUCTIONS

Math instruction pop the TOP item and NEXT to top item of the Parameter Stack 74 to use as the operands. The results are pushed back on the Parameter Stack. The CARRY flag is used to latch the "33rd bit" of the ALU result.

ADD—Pop the TOP item and NEXT to top item from the Parameter Stack, add the values together and push the result back on the Parameter Stack. The CARRY flag may be changed.

ADD-WITH-CARRY—Pop the TOP item and the NEXT to top item from the Parameter Stack, add the values together. If the CARRY flag is "1" increment the result. Push the ultimate result back on the Parameter Stack. The CARRY flag may be changed.

ADD-X—Pop the TOP item from the Parameter Stack and read the third item from the top of the Parameter Stack. Add the values together and push the result back on the Parameter Stack. The CARRY flag may be changed.

SUB—Pop the TOP item and NEXT to top item from the Parameter Stack. Subtract NEXT from TOP and push the result back on the Parameter Stack. The CARRY flag may be changed.

SUB-WITH-CARRY—Pop the TOP item and NEXT to top item from the Parameter Stack. Subtract NEXT from TOP. If the CARRY flag is "1" increment the result. Push the ultimate result back on the Parameter Stack. The CARRY flag may be changed.

SUB-X—

SIGNED-MULT-STEP—

UNSIGNED-MULT-STEP SIGNED-FAST-MULT—

FAST-MULT-STEP—

UNSIGNED-DIV-STEP—

GENERATE-POLYNOMIAL—

ROUND—

COMPARE—Pop the TOP item and NEXT to top item from the Parameter Stack. Subtract NEXT from TOP. If the result has the most significant bit equal to "0" (the result is positive), push the result onto the Parameter Stack. If the result has the most significant bit equal to "1" (the result is negative), push the old value of TOP onto the Parameter Stack. The CARRY flag may be affected.

SHIFT/ROTATE

SHIFT-LEFT—Shift the TOP Parameter Stack item left one bit. The CARRY flag is shifted into the least significant bit of TOP.

SHIFT-RIGHT—Shift the TOP Parameter Stack item right one bit. The least significant bit of TOP is shifted into the CARRY flag. Zero is shifted into the most significant bit of TOP.

DOUBLE-SHIFT-LEFT—Treating the TOP item of the Parameter Stack as the most significant word of a 64-bit number and the NEXT stack item as the least significant word, shift the combined 64-bit entity left one bit. The CARRY flag is shifted into the least significant bit of NEXT.

DOUBLE-SHIFT-RIGHT—Treating the TOP item of the Parameter Stack as the most significant word of a 64-bit number and the NEXT stack item as the least significant word, shift the combined 64-bit entity right one bit. The least significant bit of NEXT is shifted into the CARRY flag. Zero is shifted into the most significant bit of TOP.

OTHER INSTRUCTIONS

FLUSH-STACK—Empty all on-chip Parameter Stack locations into off-chip RAM. (This instruction is useful for multitasking applications). This instruction accesses a counter which holds the depth of the on-chip stack and can require from none to 16 external memory cycles.

FLUSH-RSTACK—Empty all on-chip Return Stack locations into off-chip RAM. (This instruction is useful for multitasking applications). This instruction accesses a counter which holds the depth of the on-chip Return Stack and can require from none to 16 external memory cycles.

It should further be apparent to those skilled in the art that various changes in form and details of the invention as shown and described may be made. It is intended that such changes be included within the spirit and scope of the claims appended hereto.

What is claimed is:

1. A microprocessor system comprising:

a central processing unit;

memory;

a bus connecting said central processing unit to said memory;

instruction fetching means that are connected to said bus to fetch instruction groups via said bus from said memory, certain of said instruction groups including at least one instruction that when executed, causes an access to an operand or an instruction or both, said operand or instruction being located a predetermined position from a boundary of said instruction groups;

an instruction register for receiving sequential instructions from a first of said instruction groups from said instruction fetching means. said first of said instruction groups including said at least one instruction;

instruction decoding means having means for generating a counter control signal and an operand control signal;

a counter that is connected to receive said counter control signal from said instruction decoding means;

operand selection means that is responsive to said operand control signal from said instruction decoding means;

instruction supplying means, responsive to said counter to select said predetermined position for supplying, in succession from said instruction register said sequential instructions to said central processing unit;

said instruction supplying means being further responsive to said counter and said operand selection means for selecting and supplying operand from said predetermined position in said instruction groups to said central processing unit;

said instruction decoding means providing said counter control signal and said operand control signal to cause said instruction supplying means to select from said instruction groups said operand or instruction or both associated with one of said instructions from said first of said instruction groups.

2. The microprocessor system of claim 1 wherein said instruction decoding means further includes means, responsive to a SKIP instruction in said instruction register for configuring said instruction fetching means such that the next instruction group is supplied to the instruction register, and for configuring said instruction supplying means to supply in succession from said instruction register said sequential instructions, beginning with the first instruction in said instruction register from said next instruction group to said central processing unit. and in which said means for generating counter control signal. also in response to the SKIP instruction supplies the counter control signal to reset said counter to zero.

3. The microprocessor system of claim 2 further comprising:

means for determining whether a predefined condition exists within said microprocessor system. and

means for controlling response of said instruction decoding means to said SKIP instruction and said predefined condition to execute or not execute said SKIP instruction based on existence of said predefined condition.

4. The microprocessor system of claim 1 further comprising:

a loop counter that is connected to receive a decrement control signal from said instruction decoding means, said instruction decoding means further including means. responsive to a MICROLOOP instruction in said instruction register, configured to supply said decrement control signal to said loop counter. said instruction supplying means being configured to supply from said instruction register beginning with the first instruction in said instruction register from said first of said instruction groups to said central processing unit and in which said means for generating the counter control signal, also in response to the MICROLOOP instruction, supplies the counter control signal for resetting said counter to zero.

5. The microprocessor system of claim 4 further comprising:

means for determining whether a predefined condition exists within said microprocessor system. and

means for controlling response of said instruction decoding means to said MICROLOOP instruction and said predefined condition to execute or not execute said MICROLOOP instruction based on existence of said predefined condition.

6. The microprocessor system of claim 1 wherein said instruction decoding means includes means for supplying control signals to said instruction fetching means such that a subsequent one of said instruction groups is supplied to said instruction register, and for configuring said instruction supplying means to supply to said central processing unit a remainder of said first of said instruction groups as said operand.

7. The microprocessor system of claim 6 wherein said instruction decoding means are configured to supply control signals to said instruction fetching means such that a subsequent one of said instruction groups supplied to said instruction register is determined in response to a branch-type instruction in said sequential instructions within said first of said instruction groups.

8. The microprocessor system of claim 1 wherein said instruction decoding means configures said instruction supplying means to supply to said central processing unit a last byte of said first of said instruction groups as said operand in response to one of said sequential instructions within said first of said instruction groups.

9. The microprocessor system of claim 1 wherein said instruction decoding means are configured to supply control signals to said instruction fetching means such that a subsequent one of said instruction groups is supplied as an operand in response to one of said sequential instructions within said first of said instruction groups.

10. The microprocessor system of claim 1 wherein said instruction decoding means are configured to supply control signals to said instruction fetching means such that a subsequent one of said instruction groups supplied to said instruction register is determined in response to a branch-type instruction in said sequential instructions within said first of said instruction groups.

11. The microprocessor system of claim 10 in which said instruction decoding means supplies said counter control signal to reset said counter in response to a branch-type instruction in said sequential instructions within said first of said instruction groups.

12. The microprocessor system of claim 10 further comprising means for determining whether a predefined condition exists within said microprocessor system. and

means for controlling response of said instruction decoding means to said branch-type instruction and said predefined condition to execute or not execute said branch-type instruction based on existence of said predefined condition.

13. The microprocessor system of claim 10 in which said instruction supplying means includes means for gating said sequential instructions within said instruction register to said central processing unit based on signals produced by said counter.

14. The microprocessor system of claim 1 wherein said instruction fetching means fetches said sequential instructions in parallel for each of said instruction groups in a single memory cycle.

15. The microprocessor system of claim 1 further comprising:

memory access testing means for testing said first of said instruction groups to determine if said sequential instructions require a memory access; and

if said memory access testing means determine a memory access is not required then supplying of control signals

to said instruction fetching means to fetch the next instruction group during the execution of a current of said instruction groups.

16. The microprocessor of claim 1 wherein said instruction supplying means includes:

a decoder connected to an output of said counter. and

a plurality of gates interposed between said instruction register and said central processing unit. said gates being controlled by signals from said decoder

17. The microprocessor of claim 1 wherein said instruction decoding means includes means for determining a width of said operand said width being related to position in said instruction register of said one of said instructions of said first of said instruction groups

18. The microprocessor of claim 1 wherein said first of said instruction groups includes a first instruction and multiple operand bytes. said instruction decoding means including means for determining a width of said operand associated with said first instruction based on position of said first instruction within said instruction register

19 The microprocessor of claim 18 wherein said instruction supplying means includes gating means for selecting one or more of said multiple operand bytes within said instruction register corresponding to said operand

20. A microprocessor comprising;

a central processing unit;

an instruction register operatively coupled to said central processing unit;

instruction fetching means for providing sequential instructions within instruction groups to said instruction register wherein certain of said instruction groups include at least one instruction that. when executed. causes an access to an operand or an instruction or both. said operand or instruction being located at a predetermined position from a boundary of said instruction groups;

instruction decoding means having a means for generating a counter control signal and an operand control signal;

a counter that is connected to receive said counter control signal from said instruction decoding means;

operand selection means that is responsive to said operand control signal from said instruction decoding means;

instruction supplying means. responsive to said counter to select said predetermined position. for successively coupling said sequential instructions of said certain of said instruction groups to said central processing unit;

said instruction supplying means being further responsive to said counter and said operand selection means for selection and supplying operands from said predetermined position in said instruction groups to said central processing unit; and

said instruction decoding means providing said counter control signal and said operand control signal to cause said instruction supplying means to select from said instruction groups said operand or instruction or both associated with particular ones of said sequential instructions.

21. The microprocessor of claim 20 wherein said instruction decoding means. upon receiving a SKIP one of said sequential instructions from a current one of said instruction groups. configures said instruction fetching means to fetch a next one of said instruction groups to said instruction register. supplies the counter control signal to reset said counter to zero and configures said instruction supplying means to supply a first one of said sequential instructions.

22. The microprocessor of claim 21 further including means for determining whether a predefined condition exists within said microprocessor system and

means for controlling response of said instruction decoding means to said SKIP instruction and said predefined condition to execute or not execute said SKIP instruction based on existence of said predefined condition

23 The microprocessor of claim 20 further comprising a loop counter, said instruction decoding means. responsive to a MICROLOOP instruction within said instruction register. providing a decrement signal to said loop counter and priding the counter control signal to reset said counter to zero. and said instruction supplying means being configured to supply from said instruction register said sequential instructions. beginning with the first instruction in said instruction register. from a current one of said instruction groups. to said central processing unit.

24 The microprocessor of claim 23 further comprising:

means for determining whether a predefined condition exists within said microprocessor system and

means for controlling response of said instruction decoding means to said MICROLOOP instruction and said predefined condition to execute or not execute said MICROLOOP instruction based on existence of said predefined condition.

25. The microprocessor of claim 20 wherein said instruction decoding means includes means. responsive to ones of said sequential instructions of predetermined type. for supplying control signals to said instruction fetching means such that a subsequent one of said instruction groups is provided to said instruction register.

26. The microprocessor of claim 25 wherein said instruction decoding means includes means for configuring said instruction supplying means to supply a remainder of a current one of said instruction groups within said instruction register as said operand to said central processing unit.

27. The microprocessor of claim 25 further comprising means for determining whether a predefined condition exists within said microprocessor system. and means for controlling response of said instruction decoding means to branch-type ones of said instructions and said predefined condition to execute or not execute said branch-type ones of said instructions based on existence of said predefined condition

28. The microprocessor of claim 20 wherein said instruction decoding means are configured to supply control signals to said instruction fetching means such that a subsequent one of said instruction groups is supplied as an operand in response to one of said sequential instructions.

29. In a microprocessor system including a central processing unit. memory. and an instruction register a method for providing instructions and operands from said memory to said central processing unit comprising the steps of:

providing instruction groups to said instruction register from said memory wherein certain of said instruction groups include at least one instruction that. when executed causes an access to an operand or an instruction or both said operand or instruction being located at a predetermined position from a boundary of said instruction groups;

decoding said at least one instruction to determine said predetermined position;

locating said predetermined position; and

supplying. from said instruction groups using the predetermined location. said operand or instruction or both to said central processing unit.

* * * * *

EXHIBIT C

TO THE DECLARATION OF JEFFREY M. FISHER ISO DEFENDANTS' REPLY ISO MOTION TO DISMISS OR, IN THE ALTERNATIVE, TO TRANSFER VENUE

FILED-CLERK
U.S. DISTRICT COURT

IN THE UNITED STATES DISTRICT COURT 2008 JUN -4 PM 2:16
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION TX EASTERN-MARSHALL

BY_______________

(1) TECHNOLOGY PROPERTIES LIMITED, INC. and (2) PATRIOT SCIENTIFIC CORPORATION, Plaintiffs, vs. (1) ACER, INC. (2) ACER AMERICA CORPORATION (3) GATEWAY, INC. Defendants.	CASE NO. 2:08CV228 TJW **Jury Trial Demanded**

COMPLAINT FOR PATENT INFRINGEMENTAND DEMAND FOR JURY TRIAL

Plaintiffs, Technology Properties Limited, Inc. ("TPL") and Patriot Scientific Corporation ("Patriot"), (collectively "Plaintiffs"), allege the following in support of their Complaint for Patent Infringement and Demand for Jury Trial ("Complaint") against Defendants, Acer, Inc, ("Acer"), Acer America Corporation ("Acer America") and Gateway, Inc. ("Gateway")

PARTIES

1. Plaintiff, Technology Properties Limited, Inc. ("TPL") is a corporation duly organized and existing under the laws of the State of California and maintains its principal place of business in San Jose, California.

2. Plaintiff, Patriot Scientific Corporation ("Patriot") is a corporation duly organized and existing under the laws of the State of Delaware and maintains its principal place

of business in Carlsbad, California

3. Upon information and belief, Defendant Acer, Inc is a Taiwan corporation with its principal place of business in Taipei, Taiwan, R.O C.

4. Upon information and belief, Defendant Acer America Corporation is a California corporation with its principal place of business in San Jose, California.

5. Upon information and belief, Defendant Gateway, Inc. is a Delaware corporation with its principal place of business in Irvine, California. Gateway is a wholly-owned subsidiary of Acer.

JURISDICTION

6. This Court has subject matter jurisdiction over this action pursuant to 28 U.S.C. §§ 1331, 1338(a) because this action arises under the patent laws of the United States, including 35 U.S.C §§ 101, *et seq* and 271, *et seq*. This Court has personal jurisdiction over Defendants because they each infringe Plaintiffs' patent by offering on their websites infringing products to their users and/or customers who reside in, or may be found in, the Eastern District of Texas. Further, each Defendant has actually transacted business with users of their websites in the Eastern District of Texas

VENUE

7. Venue is proper in this judicial district under 28 U.S.C. §§ 1391(b) and (c) and 1400(b) because Defendants have each committed acts of infringement in this district.

GENERAL ALLEGATIONS

8. On June 25, 1996, United States Patent No. 5,530,890 ("'890 patent") entitled "High Performance, Low Cost Microprocessor" was duly and legally issued. All rights and interest in the '890 patent were assigned to Patriot Scientific Corporation. A true and correct copy of the '890 patent is attached hereto as Exhibit A.

9. TPL and Patriot are co-owners of the '890 patent. TPL has the exclusive right to enforce and license the '890 patent, and has standing to sue

COUNT 1

(Patent infringement Against Acer, Inc.)

10. Paragraphs 1-9 of the Complaint set forth above are incorporated herein by reference.

11. Upon information and belief Defendant Acer has infringed and continues to infringe under 35 U.S.C § 271 the '890 patent.

12. Acer's acts of infringement have caused damage to Plaintiffs. Under 35 U.S.C. § 284, Plaintiffs are entitled to recover from Acer the damages sustained by Plaintiffs as a result of its infringement of the '890 patent. Acer's infringement of Plaintiffs' exclusive rights under the '890 patent will continue to damage Plaintiffs' business, causing irreparable harm, for which there is no adequate remedy of law, unless enjoined by this Court under 35 U.S.C. § 283.

13. Plaintiffs allege, on information and belief, that Acer's acts of infringement were willful and deliberate.

COUNT 2

(Patent infringement Against Acer America Corporation)

14. Paragraphs 1-9 of the Complaint set forth above are incorporated herein by reference.

15. Upon information and belief Defendant Acer America has infringed and continues to infringe under 35 U.S.C. § 271 the '890 patent.

16. Acer America's acts of infringement have caused damage to Plaintiffs. Under 35 U.S.C. § 284, Plaintiffs are entitled to recover from Acer America the damages sustained by Plaintiffs as a result of its infringement of the '890 patent. Acer America's infringement of Plaintiffs' exclusive rights under the '890 patent will continue to damage Plaintiffs' business, causing irreparable harm, for which there is no adequate remedy of law, unless enjoined by this Court under 35 U.S.C. § 283.

17. Plaintiffs allege, on information and belief, that Acer America's acts of infringement were willful and deliberate

COUNT 3

(Patent infringement Against Gateway, Inc.)

18. Paragraphs 1-9 of the Complaint set forth above are incorporated herein by reference.

19. Upon information and belief Defendant Gateway has infringed and continues to infringe under 35 U.S.C. § 271 the '890patent.

20. Gateway's acts of infringement have caused damage to Plaintiffs. Under 35 U.S.C. § 284, Plaintiffs are entitled to recover from Gateway the damages sustained by Plaintiffs as a result of its infringement of the '890 patent. Gateway's infringement of

Plaintiffs' exclusive rights under the '890 patent will continue to damage Plaintiffs' business, causing irreparable harm, for which there is no adequate remedy of law, unless enjoined by this Court under 35 U.S.C. § 283.

21. Plaintiffs allege, on information and belief, that Gateway's acts of infringement were willful and deliberate.

PRAYER FOR RELIEF

WHEREFORE, Plaintiffs respectfully request that this Court enter judgment against Defendants as follows:

A. For judgment that Defendants Acer, Inc., Acer America Corporation, and Gateway, Inc. have infringed and continue to infringe the '890 patent;

B. For permanent injunctions under 35 U.S.C. § 283 against Defendants and their directors, officers, employees, agents, subsidiaries, parents, attorneys, and all persons acting in concert, on behalf of, in joint venture, or in partnership with Defendants from further acts of infringement;

C. For damages to be paid by Defendants adequate to compensate Plaintiffs for their infringement, including interests, costs and disbursements as the Court may deem appropriate under 35 U.S.C. § 284;

D. For judgment finding that Defendants infringement was willful and deliberate, entitling Plaintiffs to increased damages under 35 U.S.C. § 284;

E. For judgment finding this to be an exceptional case against Defendants and awarding Plaintiffs attorney fees under 35 U.S.C. § 285; and,

F. For such other and further relief at law and in equity as the court may deem just and proper.

DEMAND FOR JURY TRIAL

Pursuant to the Federal Rules of Civil Procedure Rule 38, Plaintiffs hereby demand a jury trial on all issues triable by jury.

Dated: June 4, 2008

Respectfully submitted,

By: ______________________

S Calvin Capshaw
State Bar No 03783900
Email: ccapshaw@capshawlaw.com
Elizabeth L. DeRieux
State Bar No. 05770585
Email: ederieux@capshawlaw.com
N. Claire Abernathy
State Bar No 24053063
E-mail: chenry@capshawlaw.com
Capshaw DeRieux, LLP
1127 Judson Road, Suite 220
Longview, TX 75601
Telephone: (903) 236-9800
Facsimile: (903) 236-8787

John L. Cooper
California Bar No. 50324
Email: jcooper@fbm.com
Jeffrey M. Fisher
California Bar No. 155284
Email: jfisher@fbm.com
Helen Dutton
California Bar No. 235558
hdutton@fbm com
Farella Braun + Martel, LLP
Russ Building
235 Montgomery Street
San Francisco, CA 94104
Telephone: (415) 954-4400
Facsimile: (415) 954-4480

ATTORNEYS FOR PLAINTIFF
TECHNOLOGY PROPERTIES LIMITED

By: R Chris Bunt by permission NCA

Robert M. Parker
State Bar No 15498000
Email: rmparker@pbatyler.com
Robert Christopher Bunt
State Bar No. 00787165
Email: rcbunt@pbatyler.com
Parker, Bunt & Ainsworth, P.C.
100 East Ferguson, Ste. 1114
Tyler, TX 75702
Telephone: (903) 531-3535
Facsimile: (903) 533-9687

Charles T. Hoge
California Bar No. 110696
Email: choge@knlh.com
Kirby Noonan Lance & Hoge, LLP
350 Tenth Avenue, Suite 1300
San Diego, CA 92101
Telephone: (619) 231-8666
Facsimile: (619) 231-9593

ATTORNEYS FOR PLAINTIFF
PATRIOT SCIENTIFIC CORPORATION

US005530890A

United States Patent [19]

Moore et al.

[11] **Patent Number: 5,530,890**

[45] **Date of Patent: Jun. 25, 1996**

[54] **HIGH PERFORMANCE, LOW COST MICROPROCESSOR**

[75] Inventors: **Charles H. Moore**, Woodside; **Russell H. Fish, III**, Mt. View, both of Calif.

[73] Assignee: **Nanotronics Corporation**, Eagle Point, Oreg.

[21] Appl. No.: **480,206**

[22] Filed: **Jun. 7, 1995**

Related U.S. Application Data

[62] Division of Ser. No. 389,334, Aug. 3, 1989, Pat. No. 5,440,749.

[51] **Int. Cl.**[6] **G06F 9/22**

[52] **U.S. Cl.** **395/800**; 364/931; 364/925.6; 364/937.1; 364/965.4; 364/232.8; 364/244.3

[58] **Field of Search** 395/375, 500, 395/775, 800

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,603,934	9/1971	Heath	395/181
4,003,033	1/1977	O'Keefe et al.	395/287
4,037,090	7/1977	Raymond	364/706
4,042,972	8/1977	Grunes et al.	395/375
4,050,058	9/1977	Garlic	395/800
4,067,058	1/1978	Derchak	395/740
4,079,455	3/1978	Ozga	395/800
4,110,822	8/1978	Porter	395/375
4,125,871	11/1978	Martin	395/550
4,128,873	12/1978	Lamiaux	395/183.06
4,253,785	3/1981	Chamberlin	375/375
4,354,228	10/1982	Moore et al.	395/800
4,376,977	3/1983	Brunshorst	395/375
4,382,279	5/1983	Mgon	395/800
4,403,303	9/1983	Howes et al.	395/500
4,450,519	5/1984	Guttag et al.	395/800
4,463,421	7/1984	Laws	395/325
4,538,239	8/1985	Magar	364/759

(List continued on next page.)

OTHER PUBLICATIONS

C. Whitby-Strevans, "The transputer", *The 12th Annual International Symposium on Computer Architecture Conference Proceeings*, Jun. 17–19, 1985, pp. 292–300.

D. W. Best et al., "An Advanced-Architecture CMOS/SOS Microprocessor", *IEEE Micro*, vol. 2, No. 3, Aug. 1982, pp. 11–25.

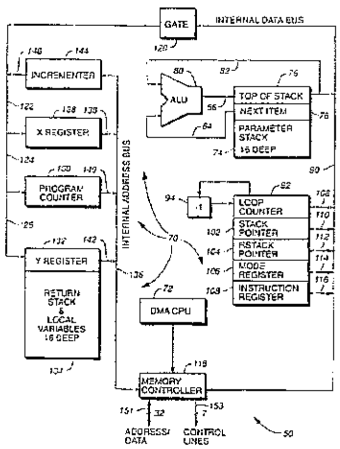
Primary Examiner—David Y. Eng

Attorney, Agent, or Firm—Cooley Godward Castro Huddleson & Tatum

[57] **ABSTRACT**

A microprocessor (**50**) includes a main central processing unit (CPU) (**70**) and a separate direct memory access (DMA) CPU (**72**) in a single integrated circuit making up the microprocessor (**50**). The main CPU (**70**) has a first 16 deep push down stack (**74**), which has a top item register (**76**) and a next item register (**78**), respectively connected to provide inputs to an arithmetic logic unit (ALU) (**80**) by lines (**82**) and (**84**). An output of the ALU (**80**) is connected to the top item register (**76**) by line (**86**). The output of the top item register at (**82**) is also connected by line (**88**) to an internal data bus (**90**). A loop counter (**92**) is connected to a decrementer (**94**) by lines (**96**) and (**98**). The loop counter (**92**) is bidirectionally connected to the internal data bus (**90**) by line (**100**). Stack pointer (**102**), return stack pointer (**104**), mode register (**106**) and instruction register (**108**) are also connected to the internal data bus (**90**) by lines (**110**), (**112**), (**114**) and (**116**), respectively. The internal data bus (**90**) is connected to memory controller (**118**) and to gate (**120**). The gate (**120**) provides inputs on lines (**122**), (**124**) and (**126**) to X register (**128**), program counter (**130**) and Y register (**132**) of return push down stack (**134**). The X register (**128**), program counter (**130**) and Y register (**132**) provide outputs to internal address bus (**136**) on lines (**138**), (**140**) and (**142**). The internal address bus provides inputs to the memory controller (**118**) and to an incrementer (**144**). The incrementer (**144**) provides inputs to the X register, program counter and Y register via lines (**146**), (**122**), (**124**) and (**126**). The DMA CPU (**72**) provides inputs to the memory controller (**118**) on line (**148**). The memory controller (**118**) is connected to a RAM by address/data bus (**150**) and control lines (**152**).

10 Claims, 19 Drawing Sheets



U S PATENT DOCUMENTS

4,541 045	9/1985	Kromer	395/375
4 562 537	12/1985	Barnett et al	395/375
4 577 282	3/1986	Candel et al	395/800
4,607,332	8/1986	Goldberg	395/375
4 626 988	12/1986	George et al	395/375
4,649 471	3/1987	Briggs	395/325
4 665,495	5/1987	Thaden	345/185
4 709 329	11/1987	Hecker	395/275
4 713 749	12/1987	Magar et al	395/375
4 714 994	12/1987	Oklobdzija et al	395/375
4 720 812	1/1988	Kao et al	395/700
4,772,888	9/1988	Kimura	340/825.5
4,777 591	10/1988	Chang et al	395/800
4 787 032	11/1988	Culley et al	395/725
4 803 621	2/1989	Kelly	395/400
4 860,198	8/1989	Takenaka	395/307
4 870,562	9/1989	Kimoto	395/550
4,931 986	6/1990	Daniel et al	395/550
5,036,460	7/1991	Takahira	395/425
5,070 451	12/1991	Moore et al	395/375
5,127 091	6/1992	Bonfarah	395/375

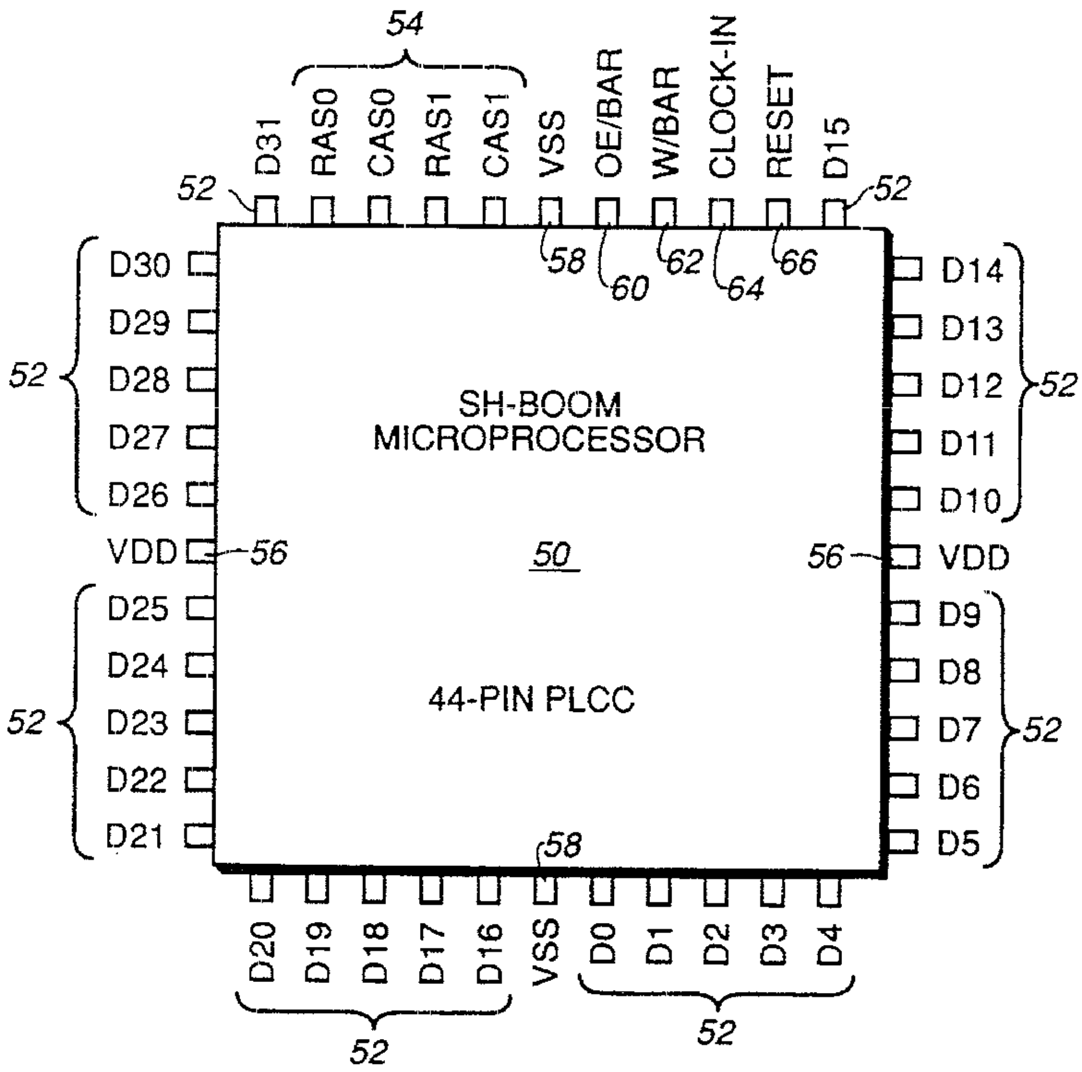
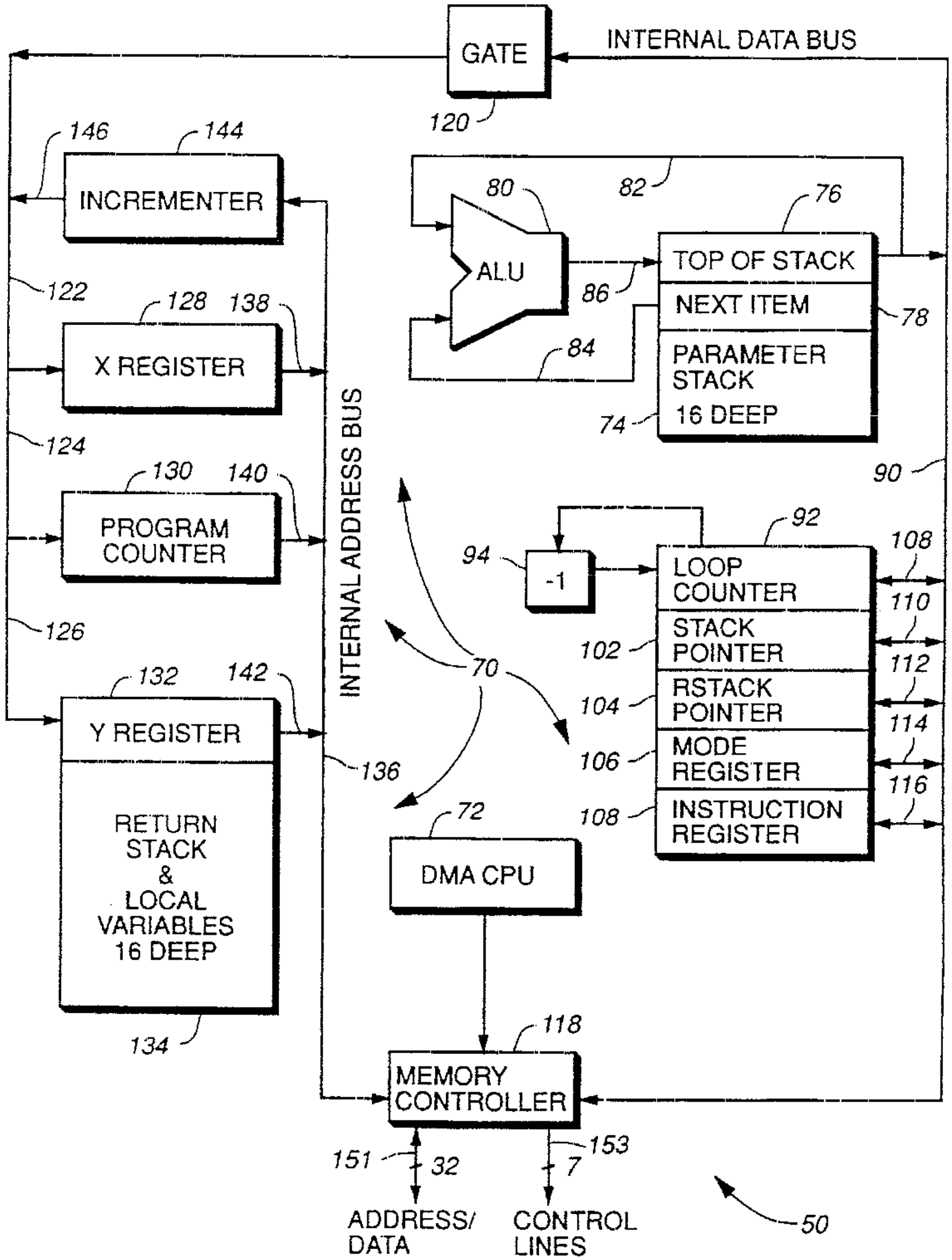
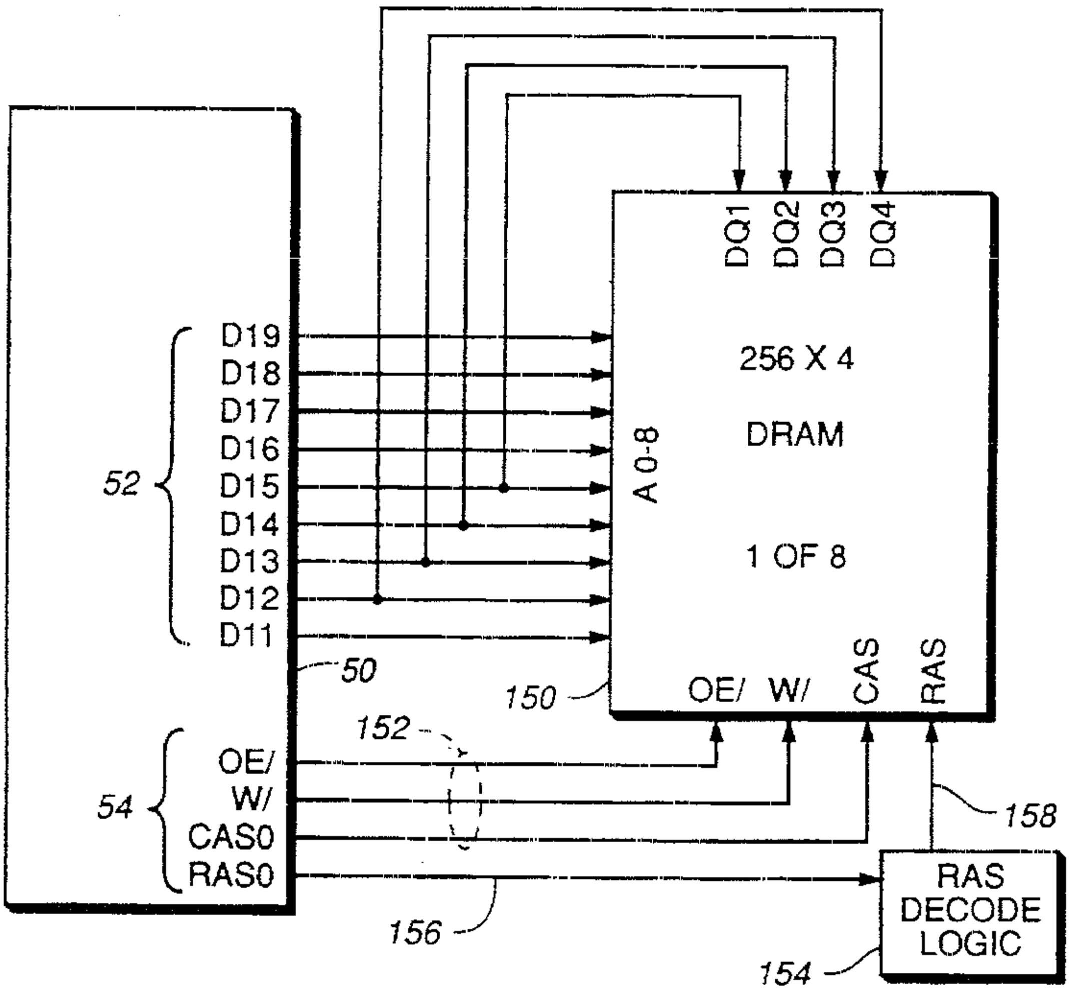

54
D31
RAS0
CAS0
RAS1
CAS1
VSS
OE/BAR
W/BAR
CLOCK-IN
RESET
D15
52
58
60
62
64
66
D30
D29
D28
D27
D26
VDD
56
D25
D24
D23
D22
D21
D14
D13
D12
D11
D10
VDD
D9
D8
D7
D6
D5
SH-BOOM
MICROPROCESSOR
50
44-PIN PLCC
D20
D19
D18
D17
D16
VSS
D0
D1
D2
D3
D4

FIG._1



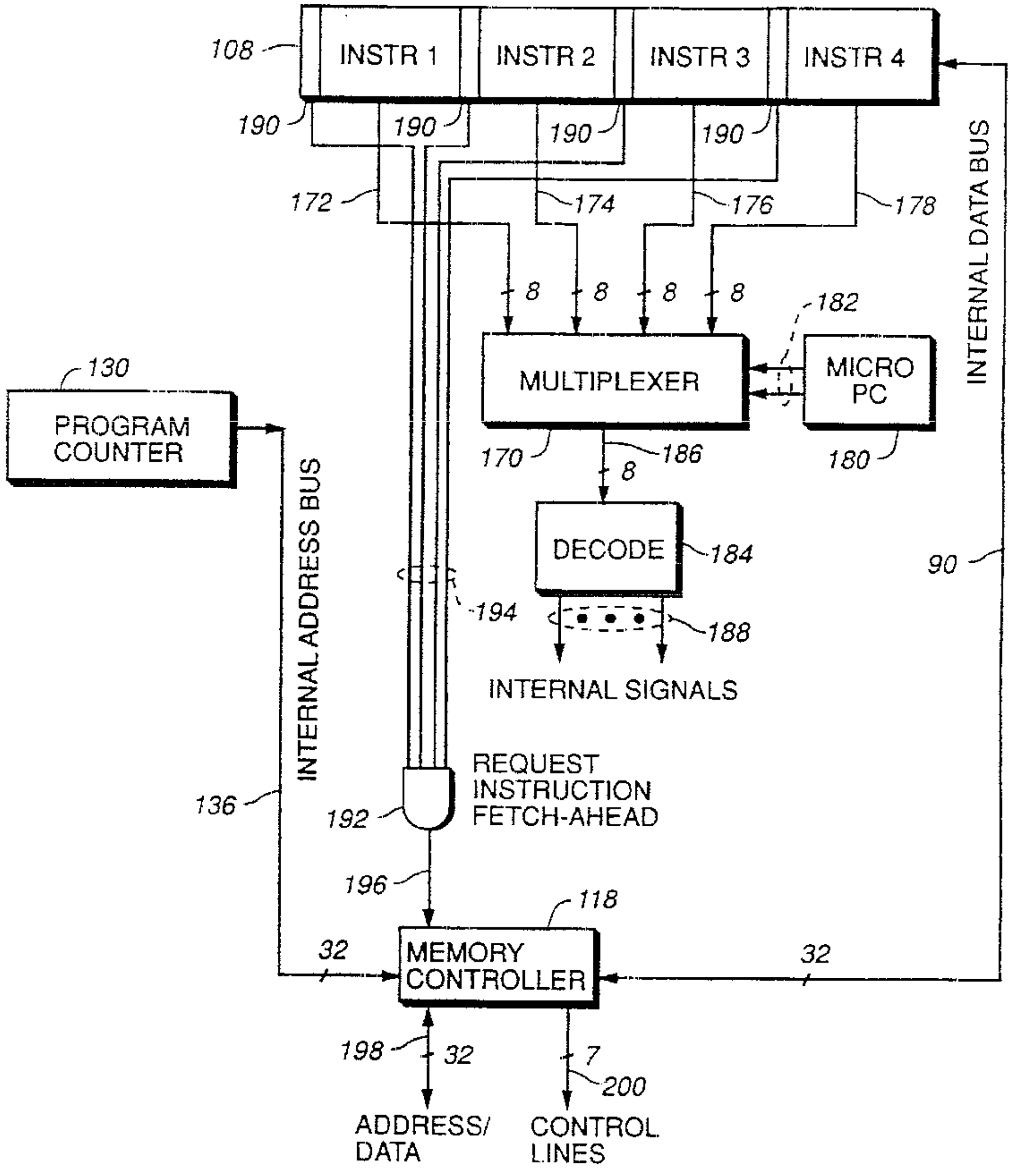
GATE
INTERNAL DATA BUS
120
146
144
INCREMENTER
80
82
76
ALU
TOP OF STACK
NEXT ITEM
PARAMETER STACK 16 DEEP
86
84
78
74
122
128
138
X REGISTER
124
130
140
PROGRAM COUNTER
INTERNAL ADDRESS BUS
90
92
94
-1
LOOP COUNTER
108
STACK POINTER
110
102
RSTACK POINTER
112
104
MODE REGISTER
114
106
INSTRUCTION REGISTER
116
108
126
70
132
142
Y REGISTER
136
RETURN STACK & LOCAL VARIABLES 16 DEEP
72
DMA CPU
134
118
MEMORY CONTROLLER
151
32
153
7
ADDRESS/ DATA
CONTROL LINES
50

FIG._2



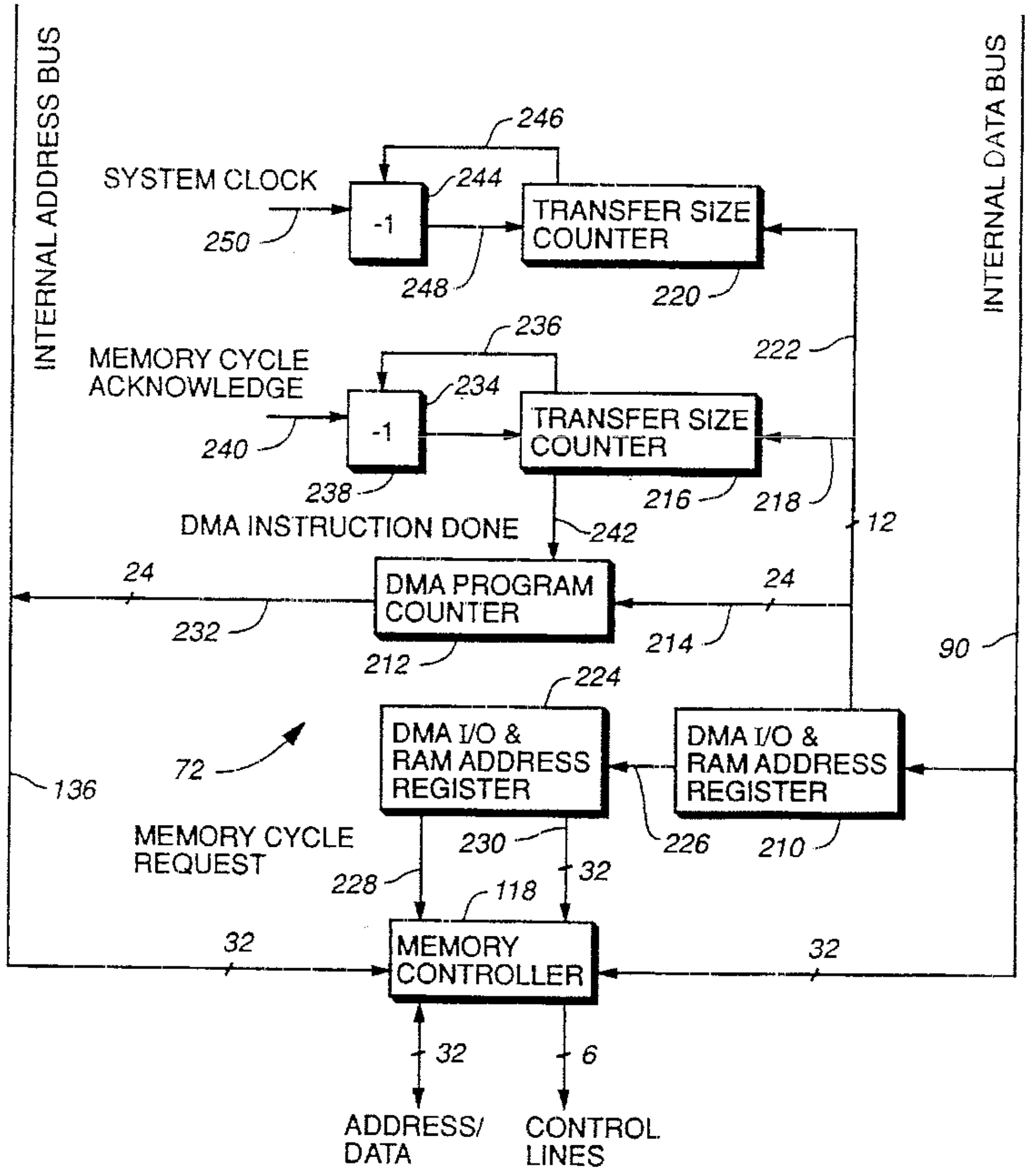
D19
D18
D17
D16
D15
D14
D13
D12
D11
52
50
54
OE/
W/
CAS0
RAS0
DQ1
DQ2
DQ3
DQ4
256 X 4
DRAM
A 0-8
1 OF 8
OE/
W/
CAS
RAS
150
152
158
156
RAS DECODE LOGIC
154

FIG._3



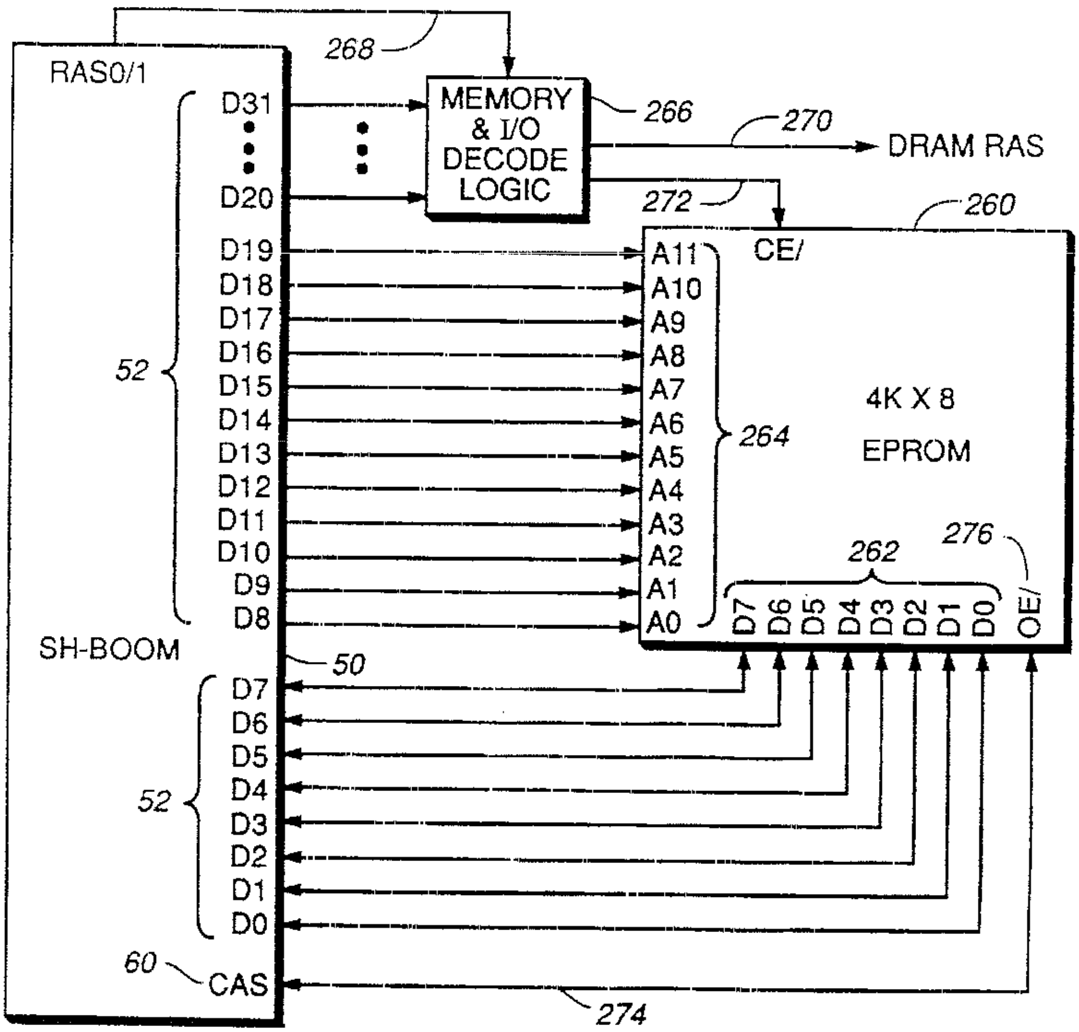
108
INSTR 1
INSTR 2
INSTR 3
INSTR 4
190
190
190
190
172
174
176
178
8
8
8
8
182
INTERNAL DATA BUS
130
PROGRAM COUNTER
MULTIPLEXER
MICRO PC
170
186
180
8
INTERNAL ADDRESS BUS
DECODE
184
90
194
188
INTERNAL SIGNALS
REQUEST INSTRUCTION FETCH-AHEAD
136
192
196
118
32
MEMORY CONTROLLER
32
198
32
7
200
ADDRESS/ DATA
CONTROL LINES

FIG._4



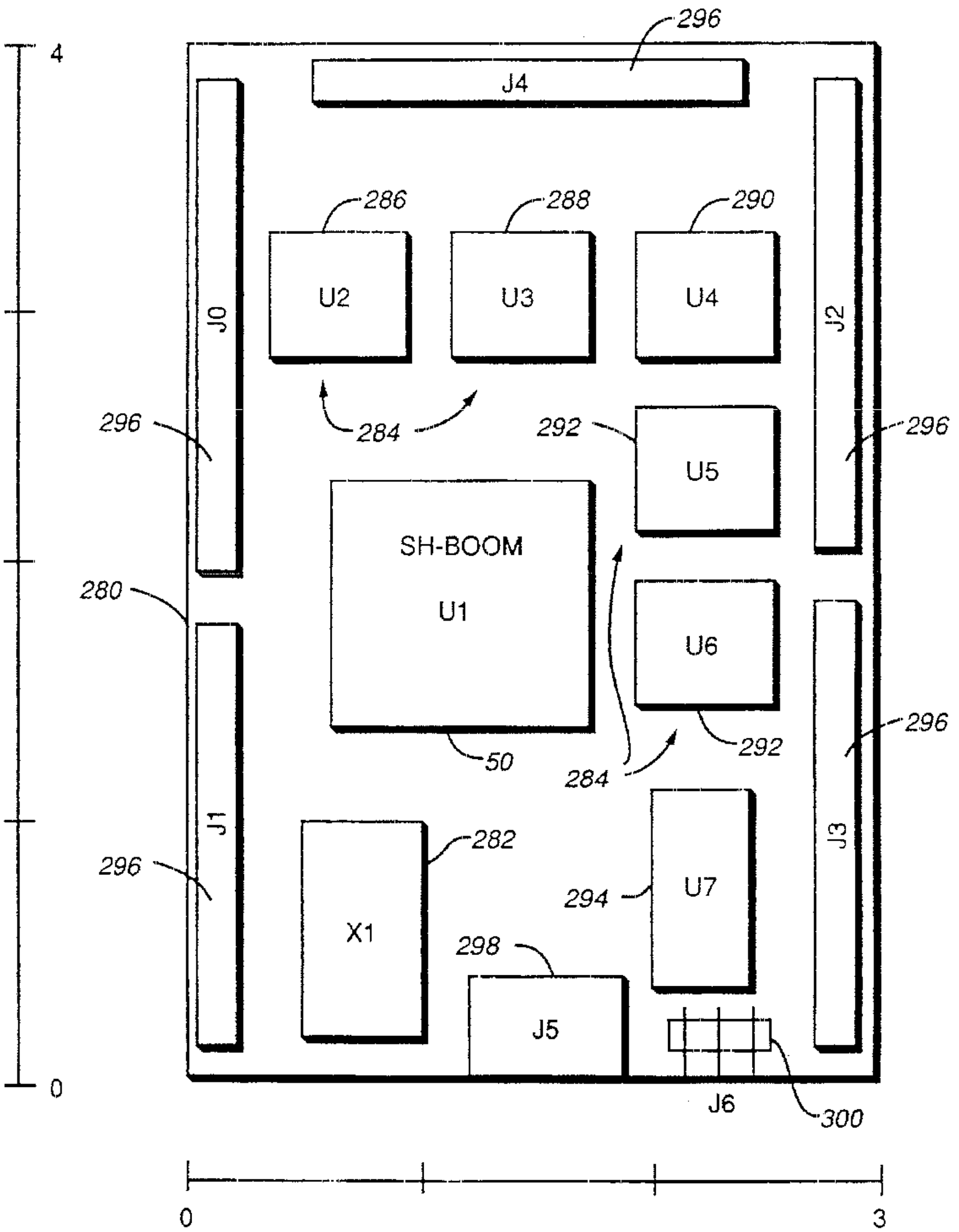
INTERNAL ADDRESS BUS
INTERNAL DATA BUS
246
SYSTEM CLOCK
244
-1
TRANSFER SIZE COUNTER
250
248
220
236
MEMORY CYCLE ACKNOWLEDGE
222
234
-1
TRANSFER SIZE COUNTER
240
238
216
218
12
DMA INSTRUCTION DONE
242
24
DMA PROGRAM COUNTER
24
232
214
90
212
224
DMA I/O & RAM ADDRESS REGISTER
DMA I/O & RAM ADDRESS REGISTER
72
136
MEMORY CYCLE REQUEST
230
226
210
32
228
118
32
MEMORY CONTROLLER
32
32
6
ADDRESS/ DATA
CONTROL LINES

FIG._5



RAS0/1
268
MEMORY & I/O DECODE LOGIC
266
270
DRAM RAS
272
260
D31
D20
D19
D18
D17
D16
D15
D14
D13
D12
D11
D10
D9
D8
52
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
CE/
264
4K X 8
EPROM
262
276
D7 D6 D5 D4 D3 D2 D1 D0 OE/
SH-BOOM
50
D7
D6
D5
D4
D3
D2
D1
D0
52
60
CAS
274

FIG._6



296
4
J4
J0
J2
286
288
290
U2
U3
U4
296
284
292
U5
296
SH-BOOM
U1
280
U6
296
50
292
284
J1
J3
296
282
X1
294
U7
298
J5
0
J6
300
0
3

FIG._7

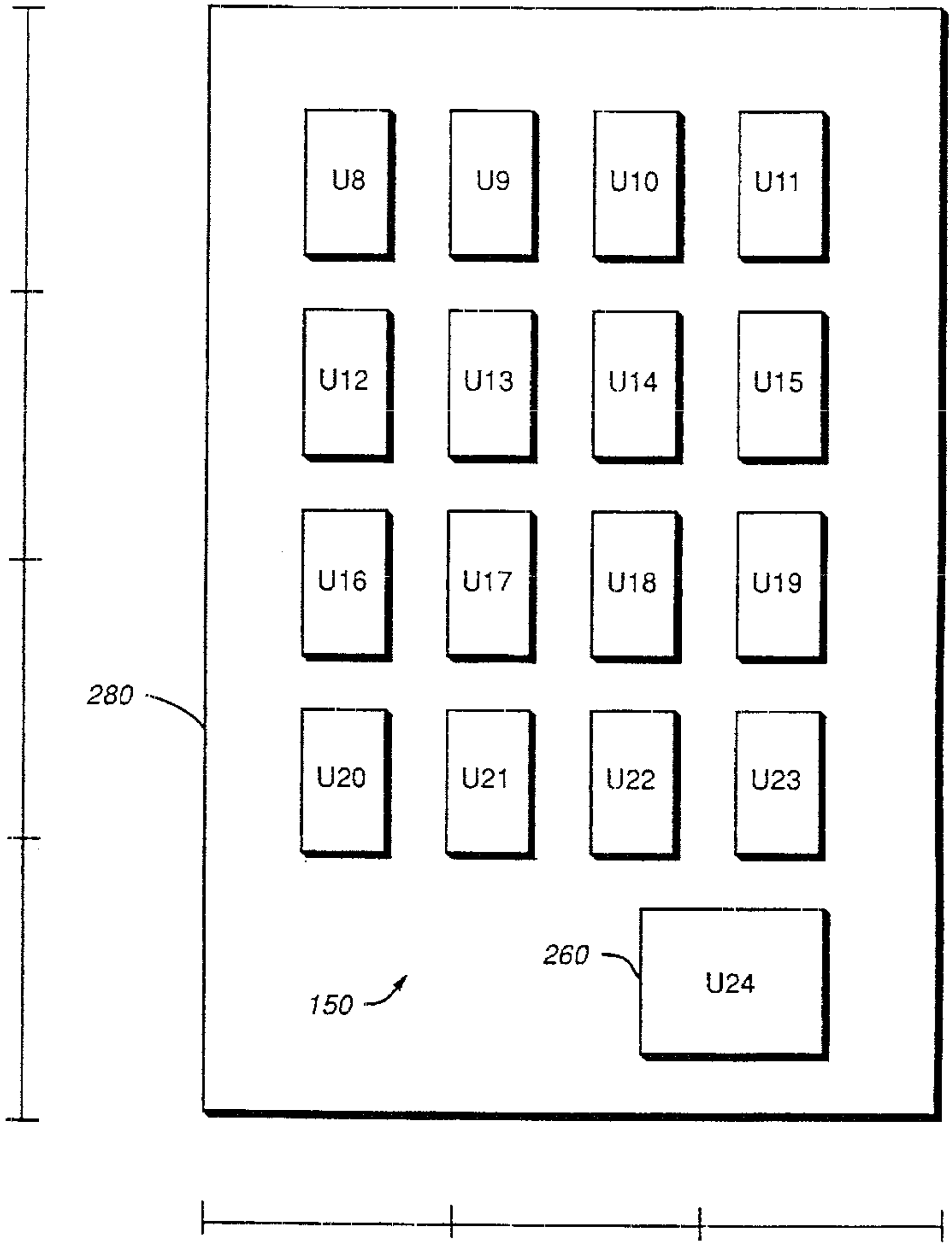
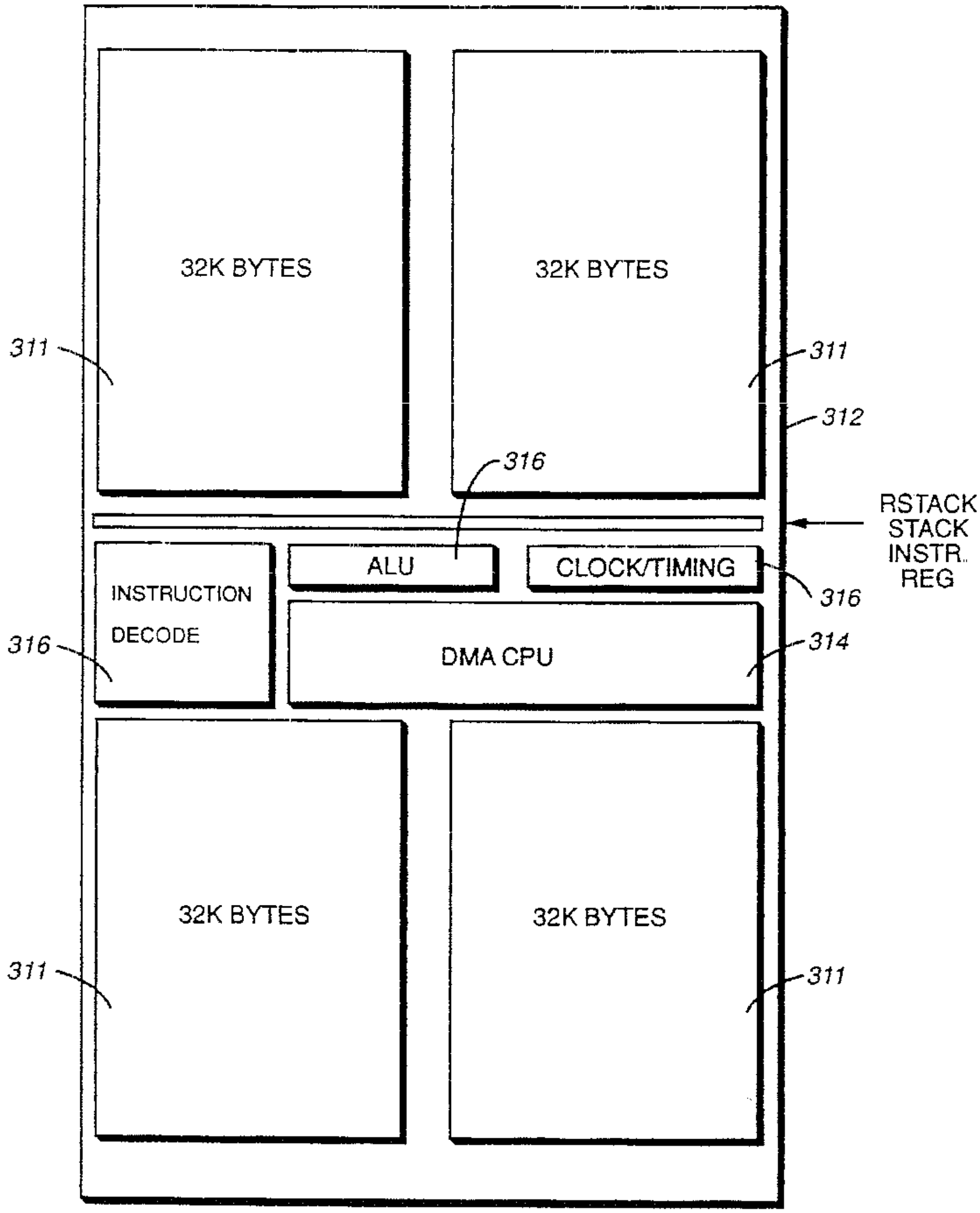

U8
U9
U10
U11
U12
U13
U14
U15
U16
U17
U18
U19
U20
U21
U22
U23
U24
280
260
150

FIG._8



32K BYTES
32K BYTES
311
311
312
316
RSTACK
STACK
INSTR.
REG
ALU
CLOCK/TIMING
INSTRUCTION
DECODE
316
316
DMA CPU
314
32K BYTES
32K BYTES
311
311

FIG._9

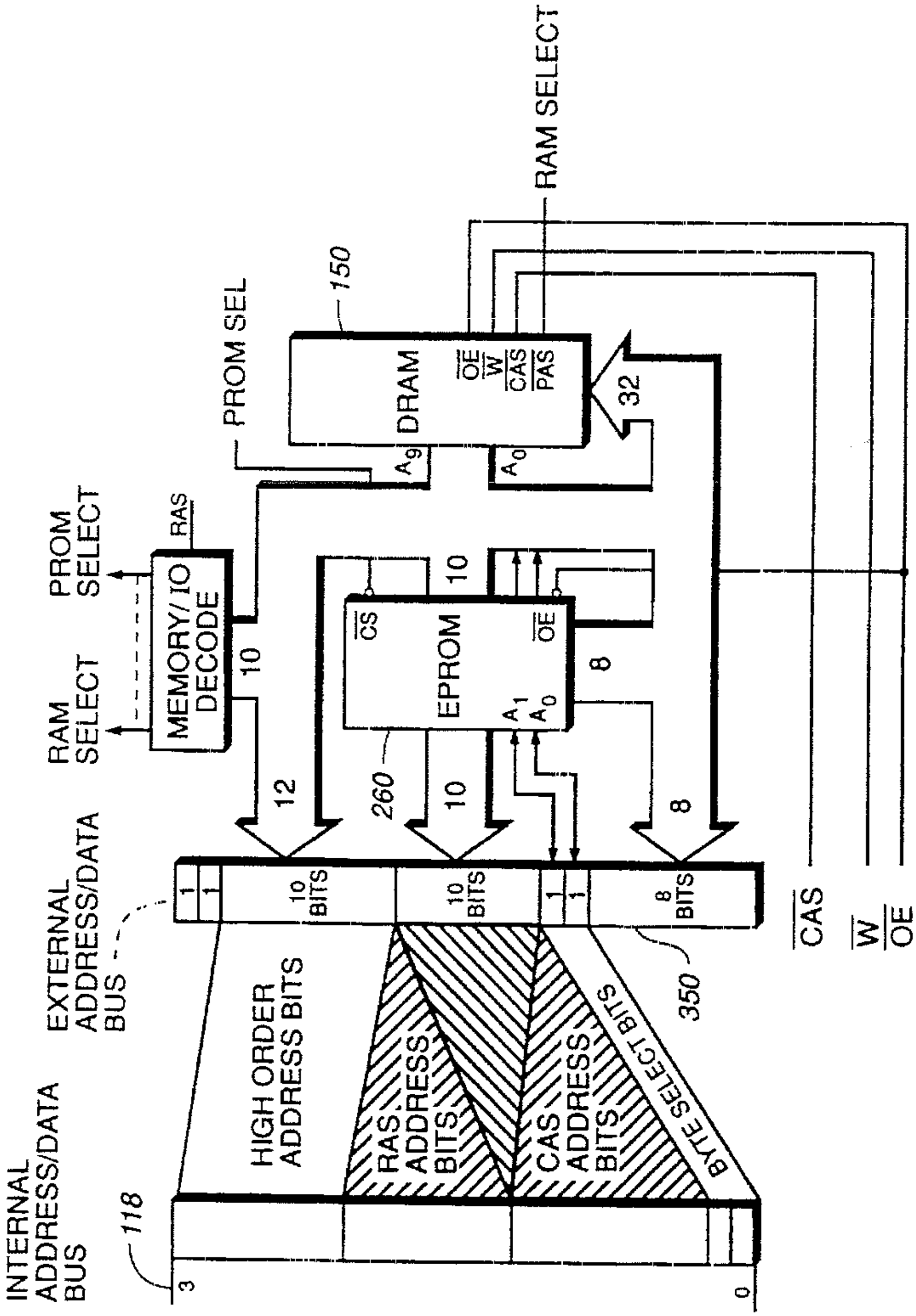

INTERNAL ADDRESS/DATA BUS
118
3
0
EXTERNAL ADDRESS/DATA BUS
HIGH ORDER ADDRESS BITS
RAS ADDRESS BITS
CAS ADDRESS BITS
BYTE SELECT BITS
350
10 BITS
10 BITS
8 BITS
1
1
RAM SELECT
PROM SELECT
MEMORY/IO DECODE
RAS
12
10
260
EPROM
CS
OE
A1
A0
8
DRAM
150
PROM SEL
A9
A0
OE
W
CAS
PAS
32
RAM SELECT
CAS
W
OE

FIG._10

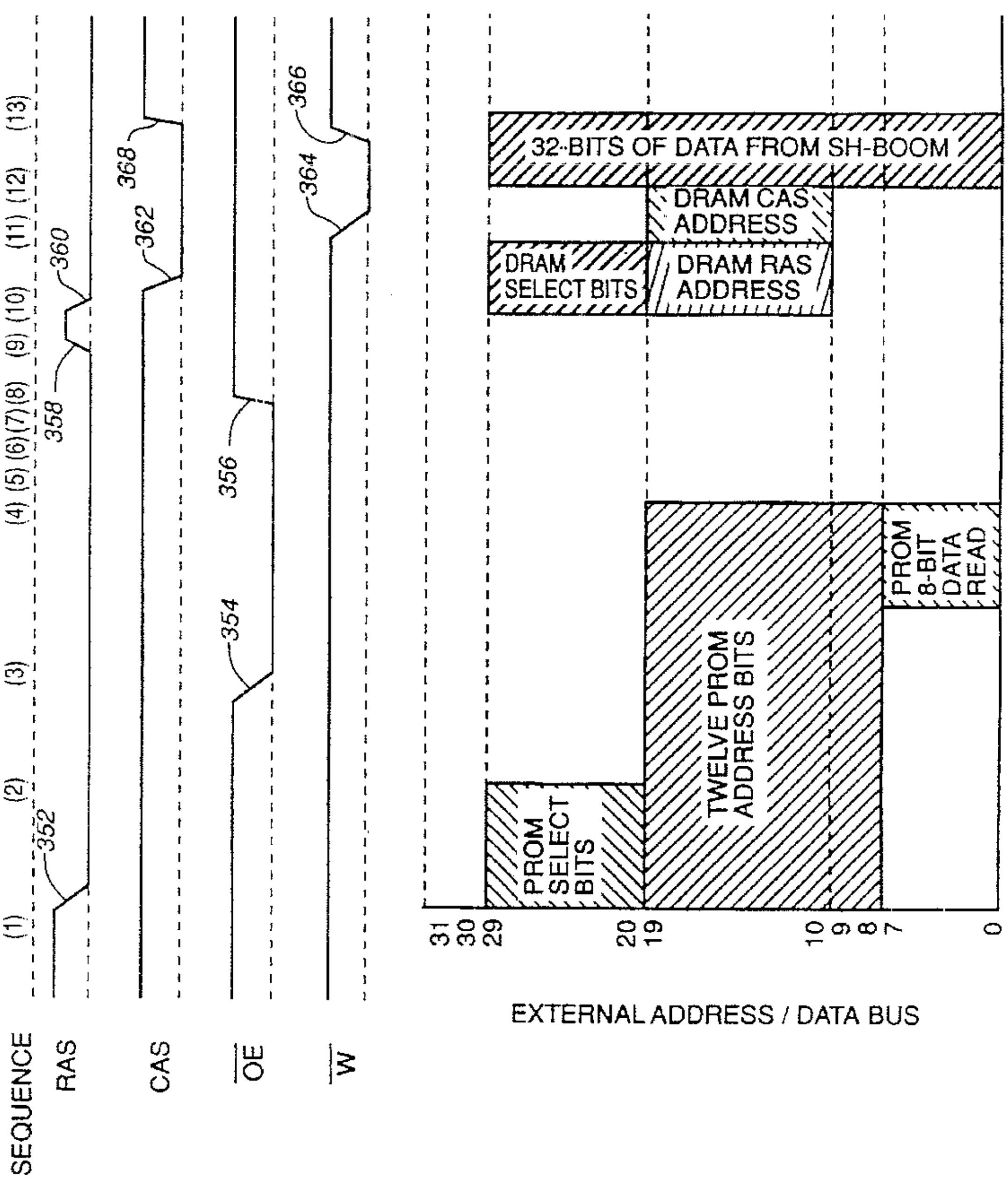

SEQUENCE
RAS
CAS
OE
W
(1)
(2)
(3)
(4) (5) (6) (7) (8)
(9) (10)
(11) (12)
(13)
352
354
356
358
360
362
364
366
368
32-BITS OF DATA FROM SH-BOOM
DRAM CAS ADDRESS
DRAM SELECT BITS
DRAM RAS ADDRESS
PROM SELECT BITS
TWELVE PROM ADDRESS BITS
PROM 8-BIT DATA READ
31
30
29
20
19
10
9
8
7
0
EXTERNAL ADDRESS / DATA BUS

FIG._11

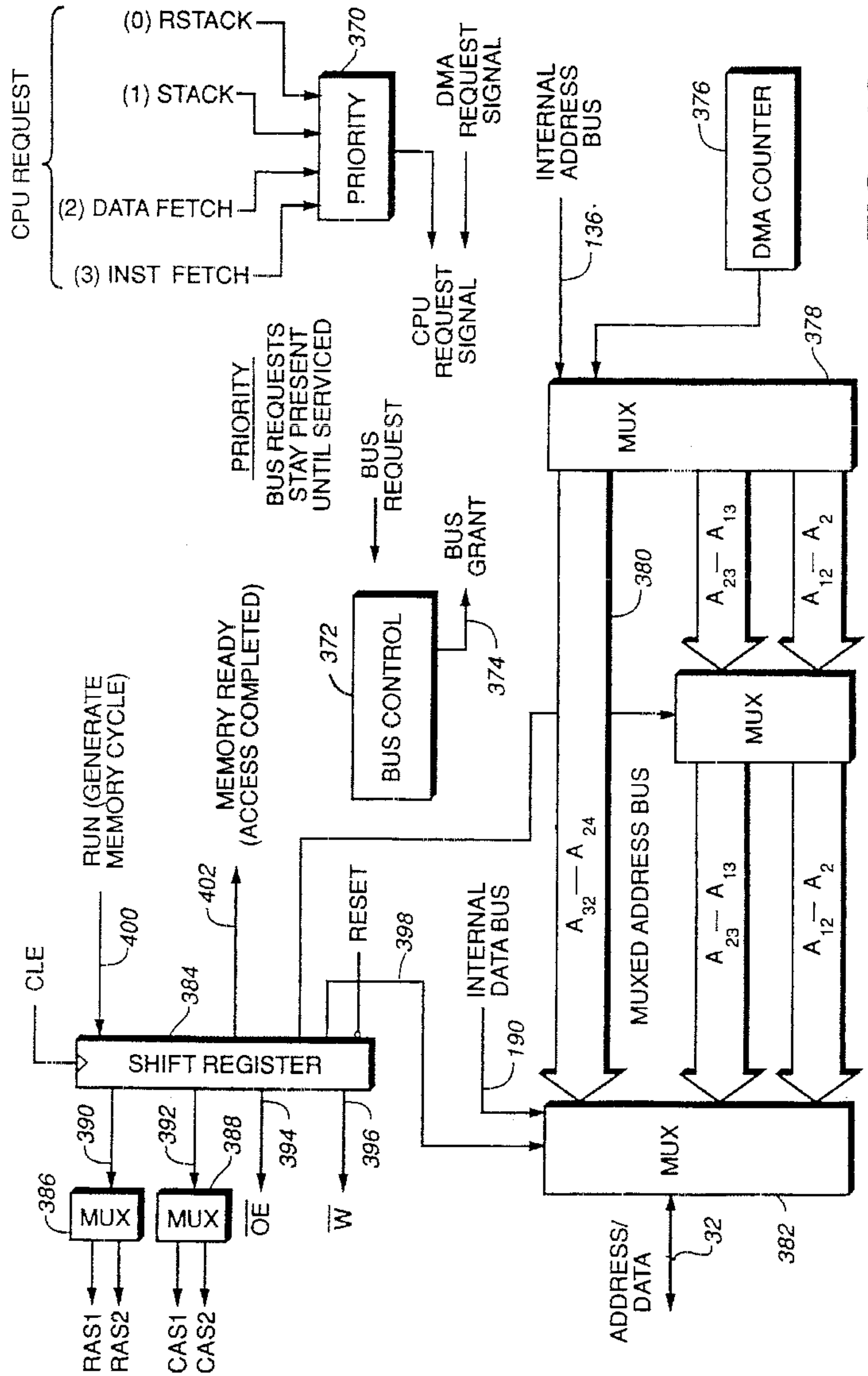

CPU REQUEST
(0) RSTACK
(1) STACK
(2) DATA FETCH
(3) INST FETCH
PRIORITY
370
DMA REQUEST SIGNAL
CPU REQUEST SIGNAL
INTERNAL ADDRESS BUS
136
376
DMA COUNTER
378
MUX
PRIORITY
BUS REQUESTS STAY PRESENT UNTIL SERVICED
BUS REQUEST
BUS GRANT
372
BUS CONTROL
374
380
A23 — A13
A12 — A2
MUX
RUN (GENERATE MEMORY CYCLE)
MEMORY READY (ACCESS COMPLETED)
400
402
CLE
384
RESET
398
SHIFT REGISTER
INTERNAL DATA BUS
190
A32 — A24
MUXED ADDRESS BUS
A23 — A13
A12 — A2
MUX
390
392
388
394
396
386
MUX
MUX
OE
W
RAS1
RAS2
CAS1
CAS2
ADDRESS/DATA
32
382

FIG._12

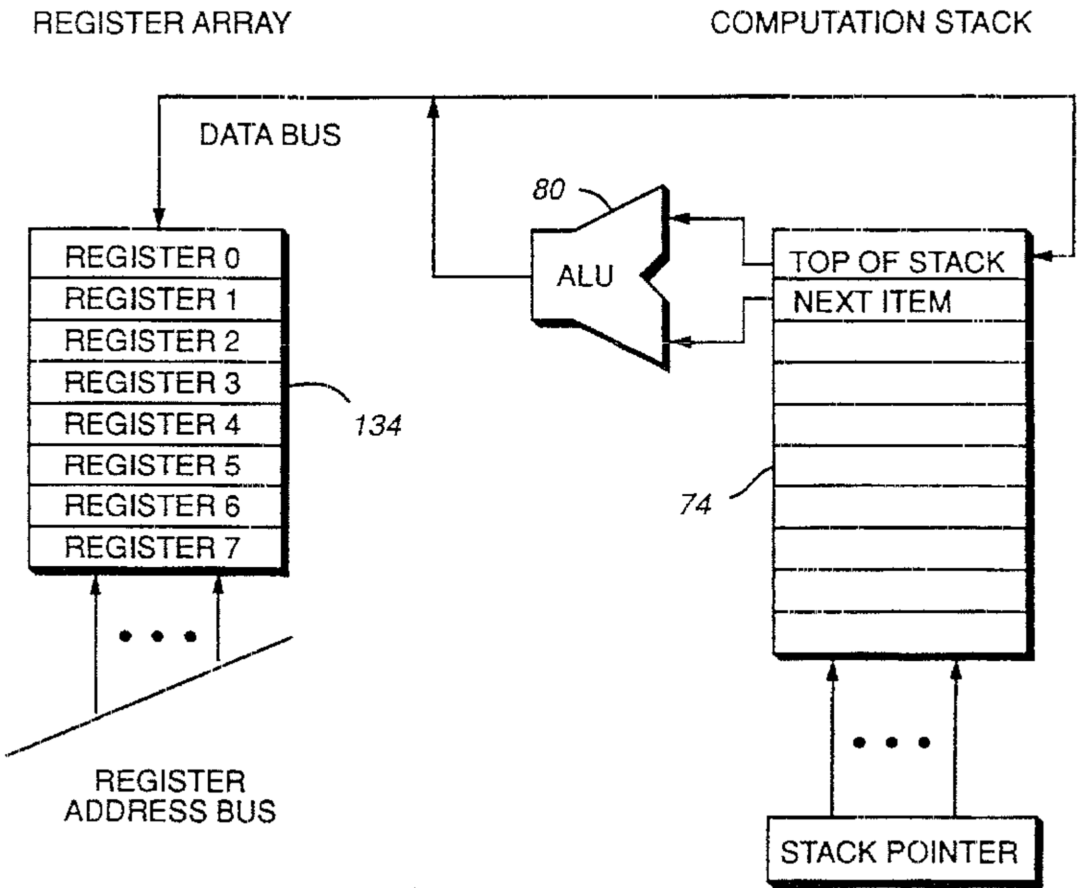

REGISTER ARRAY
COMPUTATION STACK
DATA BUS
80
ALU
TOP OF STACK
NEXT ITEM
REGISTER 0
REGISTER 1
REGISTER 2
REGISTER 3
REGISTER 4
REGISTER 5
REGISTER 6
REGISTER 7
134
74
REGISTER ADDRESS BUS
STACK POINTER

FIG._13

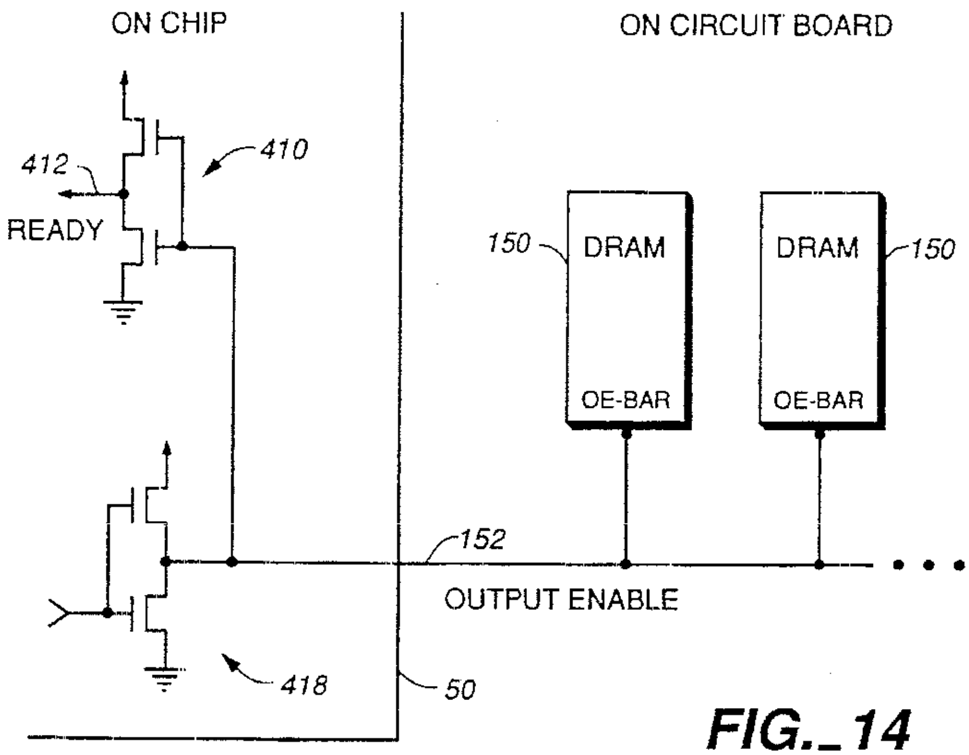

ON CHIP
ON CIRCUIT BOARD
410
412
READY
150
DRAM
DRAM
150
OE-BAR
OE-BAR
152
OUTPUT ENABLE
418
50

FIG._14

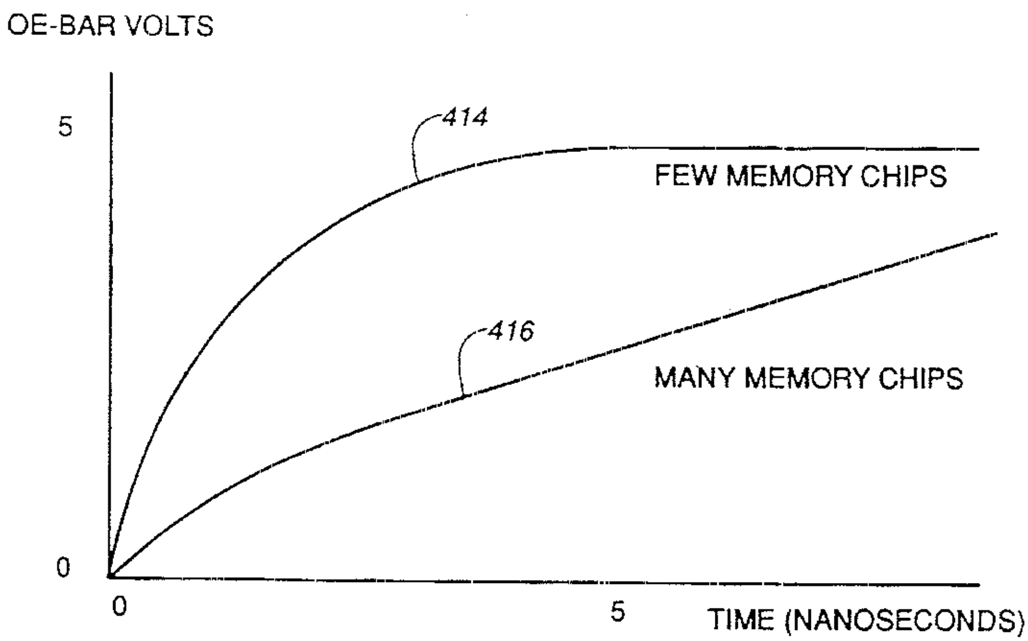

OE-BAR VOLTS
5
414
FEW MEMORY CHIPS
416
MANY MEMORY CHIPS
0
0
5
TIME (NANOSECONDS)

FIG._15

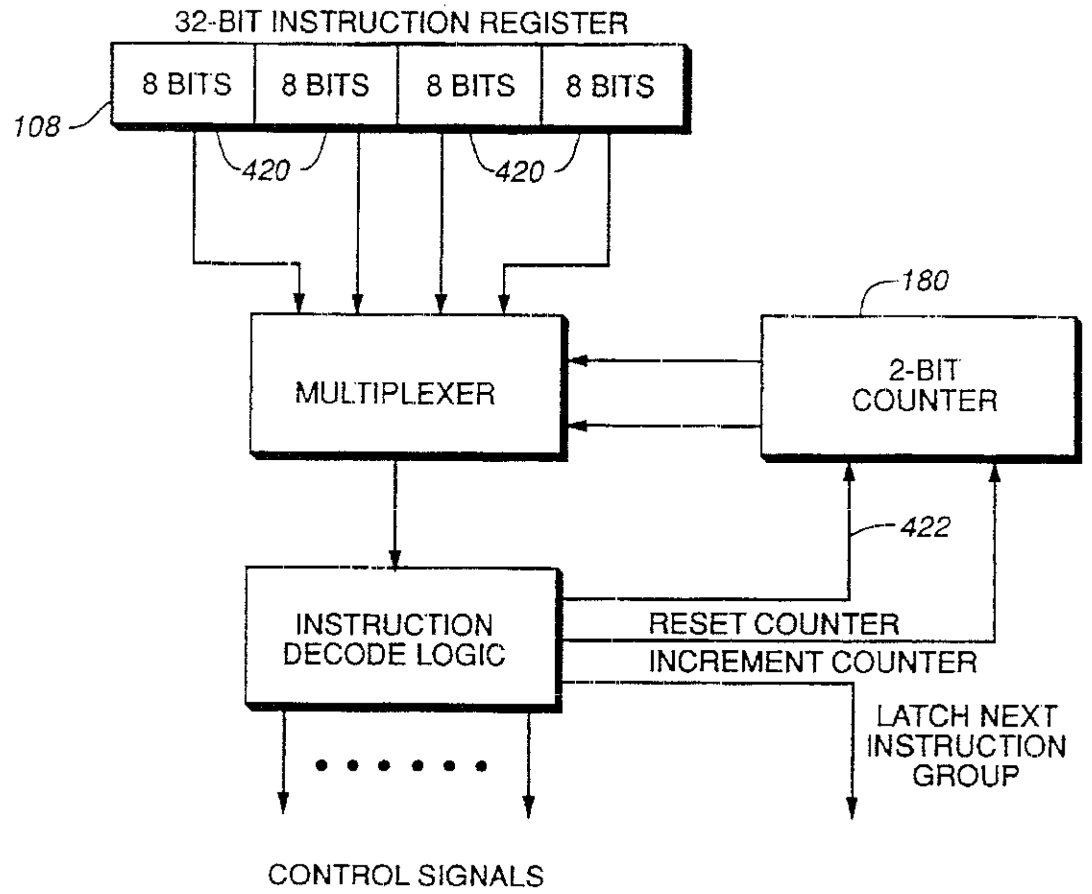

32-BIT INSTRUCTION REGISTER
8 BITS
8 BITS
8 BITS
8 BITS
108
420
420
180
MULTIPLEXER
2-BIT COUNTER
422
INSTRUCTION DECODE LOGIC
RESET COUNTER
INCREMENT COUNTER
LATCH NEXT INSTRUCTION GROUP
CONTROL SIGNALS

FIG._16

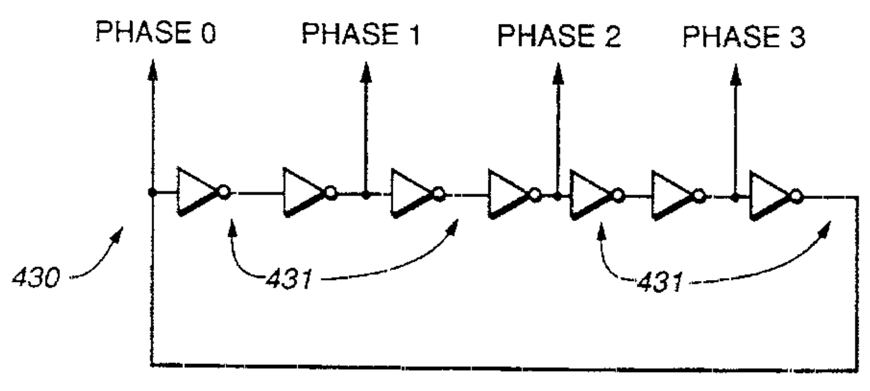

PHASE 0
PHASE 1
PHASE 2
PHASE 3
430
431
431

FIG._18

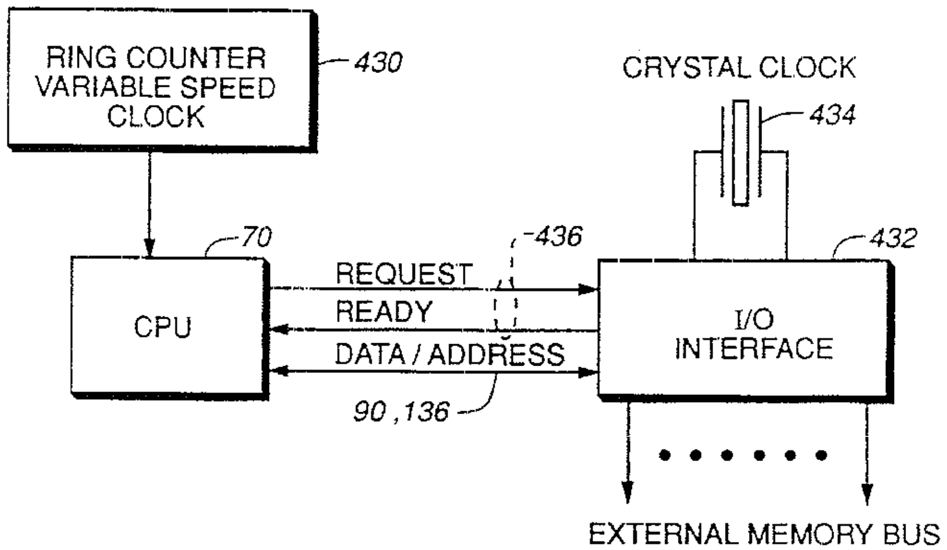

RING COUNTER VARIABLE SPEED CLOCK
430
CRYSTAL CLOCK
434
70
436
432
REQUEST
READY
DATA / ADDRESS
CPU
I/O INTERFACE
90 ,136
EXTERNAL MEMORY BUS

FIG._17

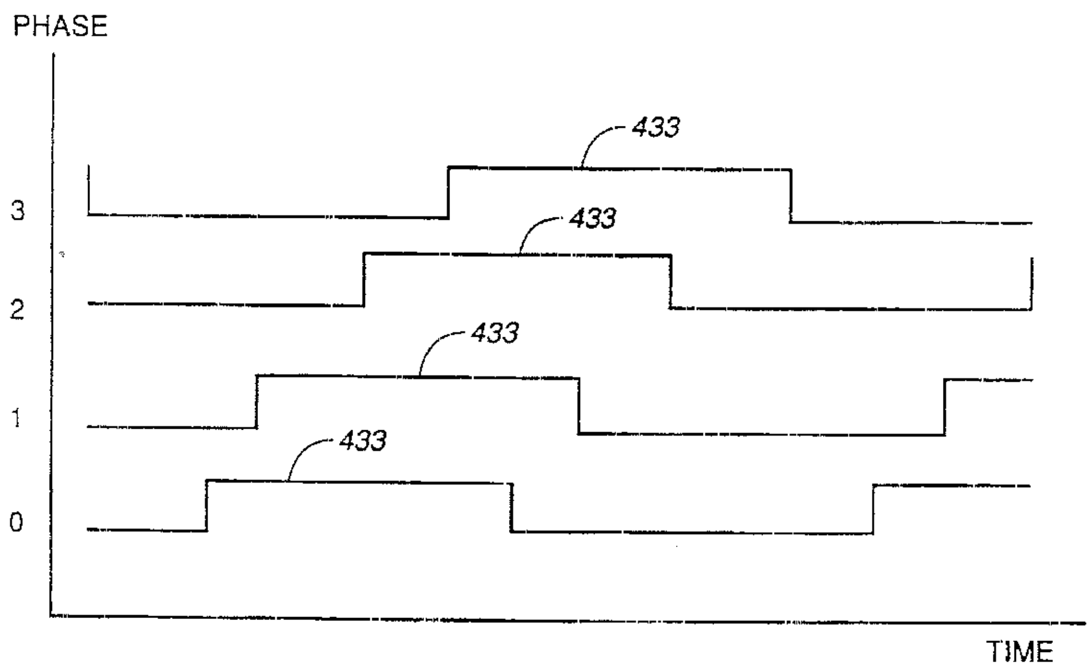

PHASE
433
433
433
433
3
2
1
0
TIME

FIG._19

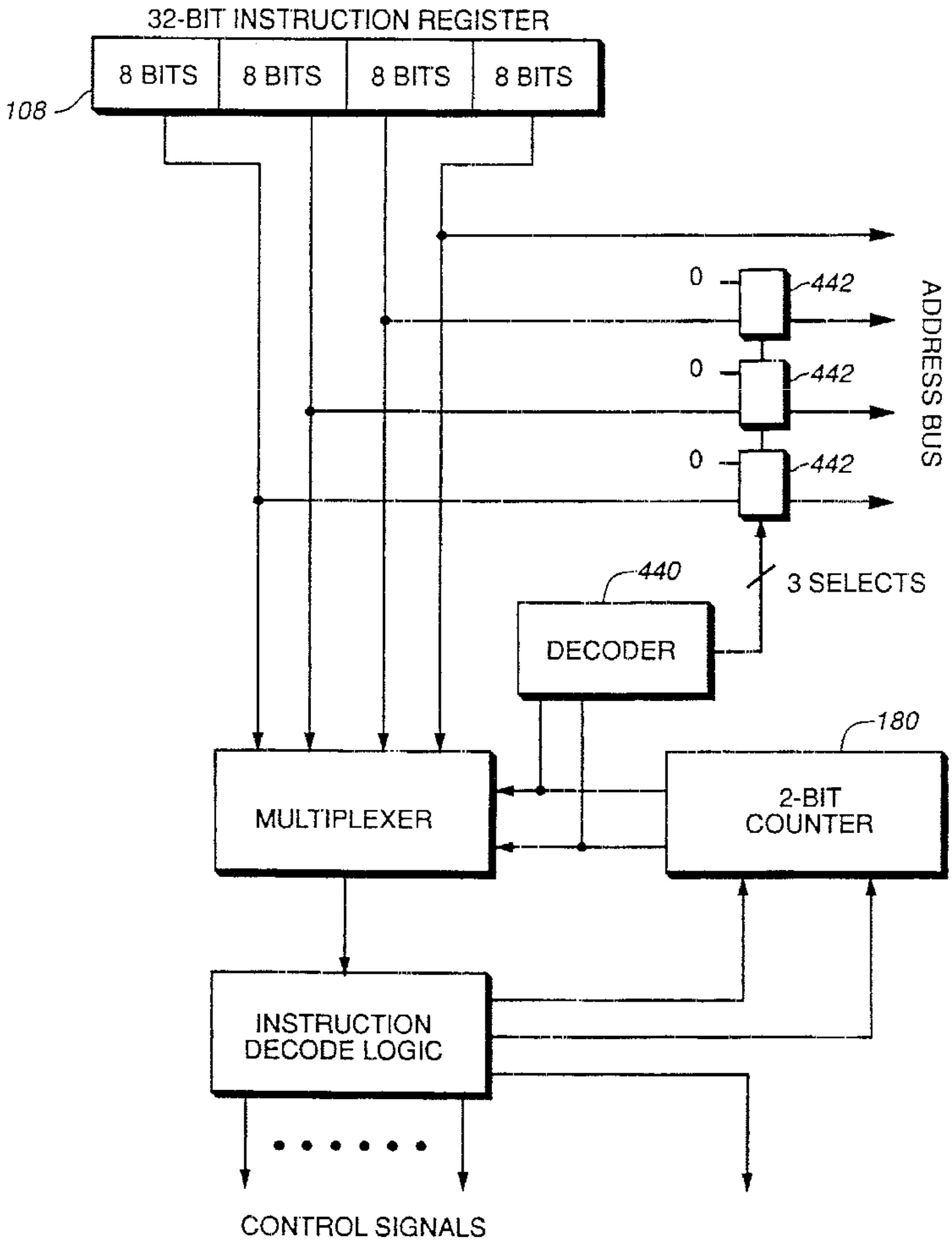
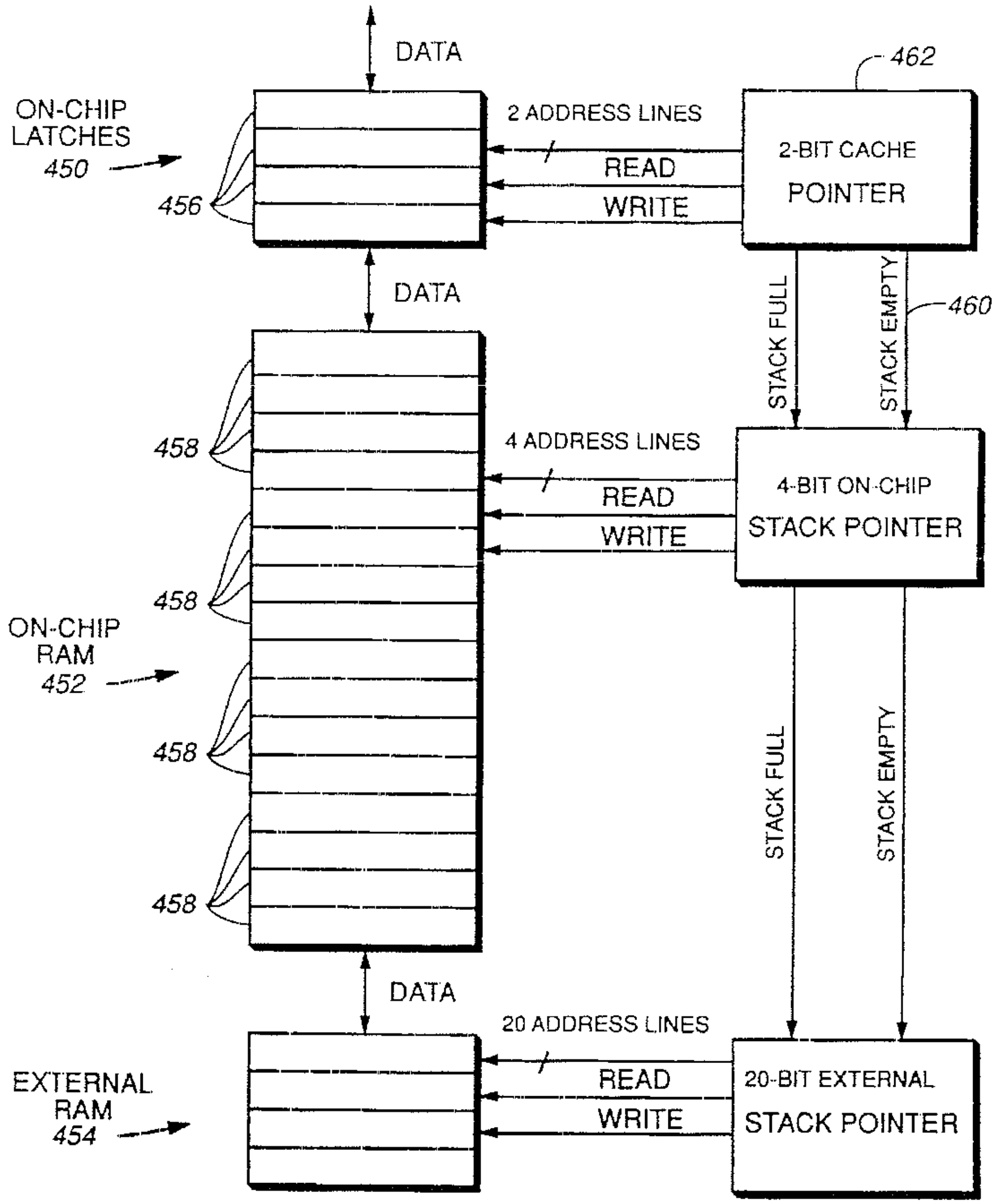
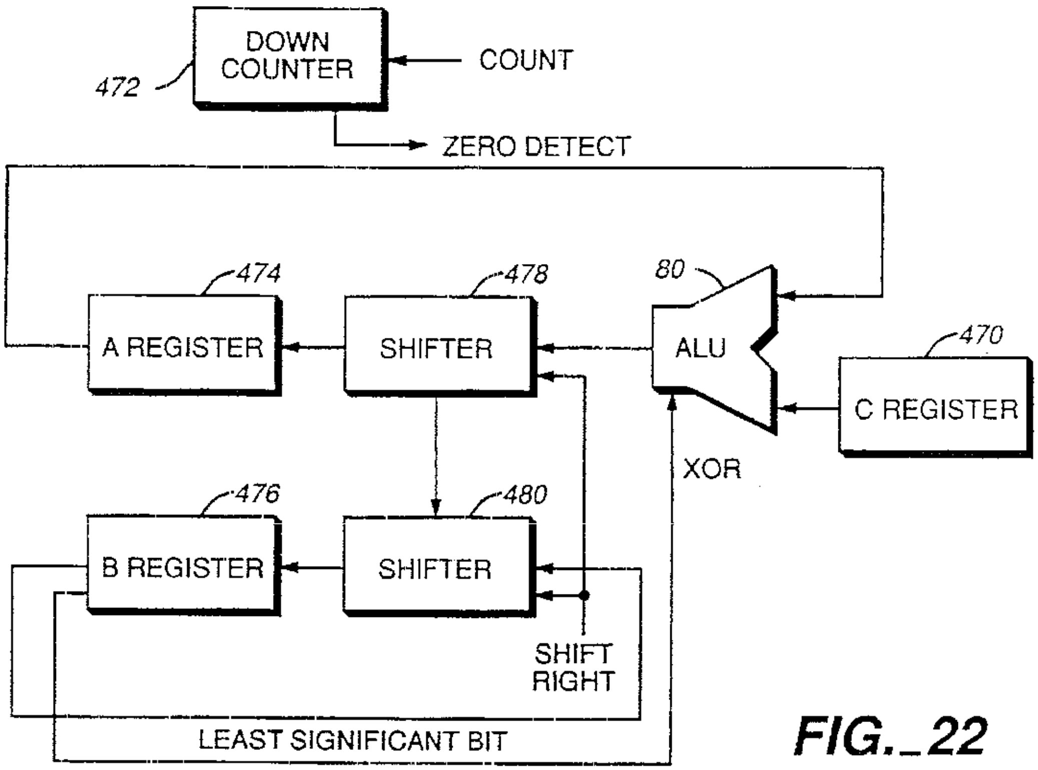

32-BIT INSTRUCTION REGISTER
8 BITS
8 BITS
8 BITS
8 BITS
108
0
442
0
442
0
442
ADDRESS BUS
440
3 SELECTS
DECODER
180
MULTIPLEXER
2-BIT COUNTER
INSTRUCTION DECODE LOGIC
CONTROL SIGNALS

FIG._20



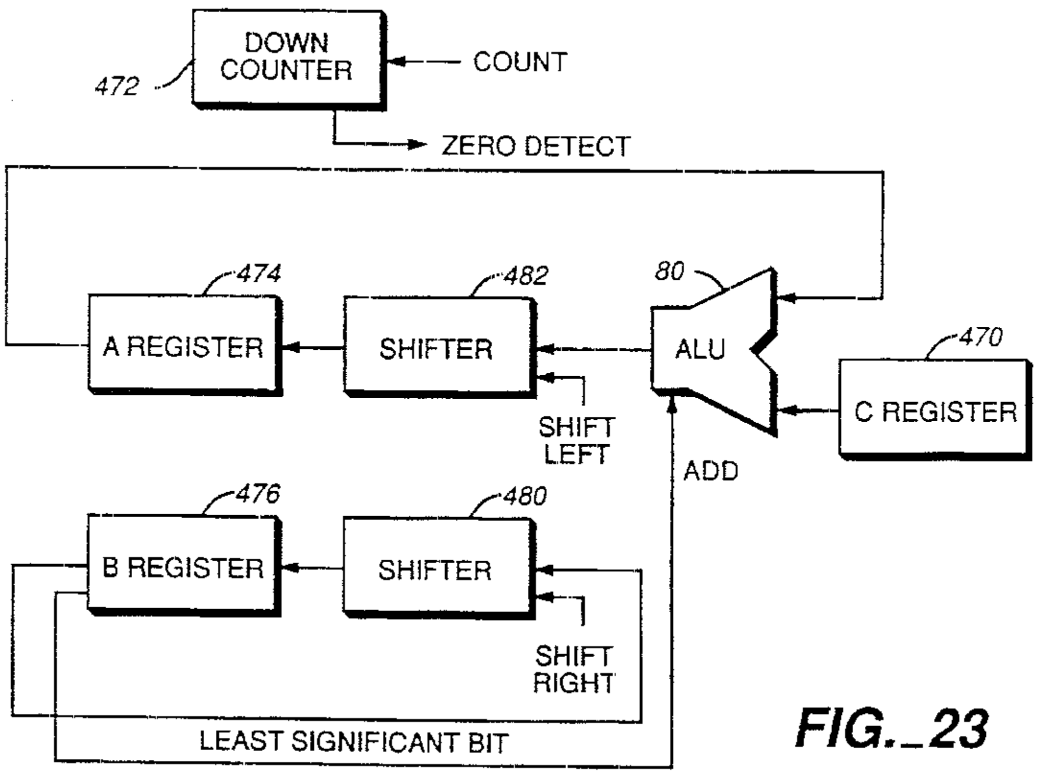
DATA
462
ON-CHIP LATCHES 450
456
2 ADDRESS LINES
READ
WRITE
2-BIT CACHE POINTER
DATA
STACK FULL
STACK EMPTY
460
458
4 ADDRESS LINES
READ
WRITE
4-BIT ON-CHIP STACK POINTER
458
ON-CHIP RAM 452
458
458
STACK FULL
STACK EMPTY
DATA
20 ADDRESS LINES
EXTERNAL RAM 454
READ
WRITE
20-BIT EXTERNAL STACK POINTER

FIG._21



DOWN COUNTER
472
COUNT
ZERO DETECT
474
A REGISTER
478
SHIFTER
80
ALU
470
C REGISTER
XOR
476
B REGISTER
480
SHIFTER
SHIFT RIGHT
LEAST SIGNIFICANT BIT

FIG._22



DOWN COUNTER
472
COUNT
ZERO DETECT
474
A REGISTER
482
SHIFTER
80
ALU
470
C REGISTER
SHIFT LEFT
ADD
476
B REGISTER
480
SHIFTER
SHIFT RIGHT
LEAST SIGNIFICANT BIT

FIG._23

HIGH PERFORMANCE, LOW COST MICROPROCESSOR

CROSS REFERENCE TO RELATED APPLICATIONS

This application is a division of U.S. application Ser No 07/389 334 filed Aug 3 1989 now U.S. Pat No. 5 440 749

BACKGROUND OF THE INVENTION

1 Field of the Invention

The present invention relates generally to a simplified reduced instruction set computer (RISC) microprocessor More particularly it relates to such a microprocessor which is capable of performance levels of for example 20 million instructions per second (MIPS) at a price of for example 20 dollars

2 Description of the Prior Art

since the invention of the microprocessor, improvements in its design have taken two different approaches In the first approach a brute force gain in performance has been achieved through the provision of greater numbers of faster transistors in the microprocessor integrated circuit and an instruction set of increased complexity. This approach is exemplified by the Motorola 68000 and Intel 80X86 microprocessor families. The trend in this approach is to larger die sizes and packages, with hundreds of pinouts

More recently, it has been perceived that performance gains can be achieved through comparative simplicity, both in the microprocessor integrated circuit itself and in its instruction set This second approach provides RISC microprocessors, and is exemplified by the Sun SPARC and the Intel 8960 microprocessors However even with this approach as conventionally practiced the packages for the microprocessor are large in order to accommodate the large number of pinouts that continue to be employed. A need therefore remains for further simplification of high performance microprocessors

With conventional high performance microprocessors fast static memories are required for direct connection to the microprocessors in order to allow memory accesses that are fast enough to keep up with the microprocessors Slower dynamic random access memories (DRAMs) are used with such microprocessors only in a hierarchical memory arrangement with the static memories acting as a buffer between the microprocessors and the DRAMs. The necessity to use static memories increases cost of the resulting systems

Conventional microprocessors provide direct memory accesses (DMA) for system peripheral units through DMA controllers which may be located on the microprocessor integrated circuit, or provided separately. Such DMA controllers can provide routine handling of DMA requests and responses, but some processing by the main central processing unit (CPU) of the microprocessor is required

SUMMARY OF THE INVENTION

Accordingly, it is an object of this invention to provide a microprocessor with a reduced pin count and cost compared to conventional microprocessors

It is another object of the invention to provide a high performance microprocessor that can be directly connected to DRAMs without sacrificing microprocessor speed.

It is a further object of the invention to provide a high performance microprocessor in which DMA does not require use of the main CPU during DMA requests and responses and which provides very rapid DMA response with predictable response times

The attainment of these and related objects may be achieved through use of the novel high performance low cost microprocessor herein disclosed. In accordance with one aspect of the invention a microprocessor system in accordance with this invention has a central processing unit a dynamic random access memory and a bus connecting the central processing unit to the dynamic random access memory There is a multiplexing means on the bus between the central processing unit and the dynamic random access memory. The multiplexing means is connected and configured to provide row addresses column addresses and data on the bus

In accordance with another aspect of the invention, the microprocessor system has a means connected to the bus for fetching instructions for the central processing unit on the bus. The means for fetching instructions is configured to fetch multiple sequential instructions in a single memory cycle In a variation of this aspect of the invention, a programmable read only memory containing instructions for the central processing unit is connected to the bus. The means for fetching instructions includes means for assembling a plurality of instructions from the programmable read only memory and storing the plurality of instructions in the dynamic random access memory

In another aspect of the invention, the microprocessor system includes a central processing unit, a direct memory access processing unit and a memory connected by a bus. The direct memory access processing unit includes means for fetching instructions for the central processing unit and for fetching instructions for the direct memory access processing unit on the bus

In a further aspect of the invention, the microprocessor system including the memory, is contained in an integrated circuit The memory is a dynamic random access memory, and the means for fetching multiple instructions includes a column latch for receiving the multiple instructions

In still another aspect of the invention, the microprocessor system additionally includes an instruction register for the multiple instructions connected to the means for fetching instructions. A means is connected to the instruction register for supplying the multiple instructions in succession from the instruction register. A counter is connected to control the means for supplying the multiple instructions to supply the multiple instructions in succession A means for decoding the multiple instructions is connected to receive the multiple instructions in succession from the means for supplying the multiple instructions. The counter is connected to said means for decoding to receive incrementing and reset control signals from the means for decoding The means for decoding is configured to supply the reset control signal to the counter and to supply a control signal to the means for fetching instructions in response to a SKIP instruction in the multiple instructions. In a modification of this aspect of the invention, the microprocessor system additionally has a loop counter connected to receive a decrement control signal from the means for decoding. The means for decoding is configured to supply the reset control signal to the counter and the decrement control signal to the loop counter in response to a MICROLOOP instruction in the multiple instructions. In a further modification to this aspect of the invention, the means for decoding is configured to control

the counter in response to an instruction utilizing a variable width operand. A means is connected to the counter to select the variable width operand in response to the counter

In a still further aspect of the invention, the microprocessor system includes an arithmetic logic unit. A first push down stack is connected to the arithmetic logic unit The first push down stack includes means for storing a top item connected to a first input of the arithmetic logic unit and means for storing a next item connected to a second input of the arithmetic logic unit. The arithmetic logic unit has an output connected to the means for storing a top item The means for storing a top item is connected to provide an input to a register file The register file desirably is a second push down stack, and the means for storing a top item and the register file are bidirectionally connected.

In another aspect of the invention, a data processing system has a microprocessor including a sensing circuit and a driver circuit a memory, and an output enable line connected between the memory, the sensing circuit and the driver circuit. The sensing circuit is configured to provide a ready signal when the output enable line reaches a predetermined electrical level, such as a voltage The microprocessor is configured so that the driver circuit provides an enabling signal on the output enable line responsive to the ready signal

In a further aspect of the invention, the microprocessor system has a ring counter variable speed system clock connected to the central processing unit. The central processing unit and the ring counter variable speed system clock are provided in a single integrated circuit. An input/output interface is connected to exchange coupling control signals, addresses and data with the input/output interface A second clock independent of the ring counter variable speed system clock is connected to the input/output interface

In yet another aspect of the invention, a push down stack is connected to the arithmetic logic unit. The push down stack includes means for storing a top item connected to a first input of the arithmetic logic unit and means for storing a next item connected to a second input of the arithmetic logic unit. The arithmetic logic unit has an output connected to the means for storing a top item. The push down stack has a first plurality of stack elements configured as latches and a second plurality of stack elements configured as a random access memory. The first and second plurality of stack elements and the central processing unit are provided in a single integrated circuit. A third plurality of stack elements is configured as a random access memory external to the single integrated circuit In this aspect of the invention, desirably a first pointer is connected to the first plurality of stack elements, a second pointer connected to the second plurality of stack elements, and a third pointer is connected to the third plurality of stack elements. The central processing unit is connected to pop items from the first plurality of stack elements. The first stack pointer is connected to the second stack pointer to pop a first plurality of items from the second plurality of stack elements when the first plurality of stack elements are empty from successive pop operations by the central processing unit. The second stack pointer is connected to the third stack pointer to pop a second plurality of items from the third plurality of stack elements when the second plurality of stack elements are empty from successive pop operations by the central processing unit

In another aspect of the invention, a first register is connected to supply a first input to the arithmetic logic unit A first shifter is connected between an output of the arithmetic logic unit and the first register. A second register is connected to receive a starting polynomial value An output of the second register is connected to a second shifter A least significant bit of the second register is connected to The arithmetic logic unit. A third register is connected to supply feedback terms of a polynomial to the arithmetic logic unit A down counter, for counting down a number corresponding to digits of a polynomial to be generated, is connected to the arithmetic logic unit. The arithmetic logic unit is responsive to a polynomial instruction to carry out an exclusive OR of the contents of the first register with the contents of the third register if the least significant bit of the second register is a "ONE" and to pass the contents of the first register unaltered if the least significant bit of the second register is a "ZERO", until the down counter completes a count The polynomial to be generated results in said first register

In still another aspect of the invention, a result register is connected to supply a first input to the arithmetic logic unit. A first, left shifting shifter is connected between an output of the arithmetic logic unit and the result register A multiplier register is connected to receive a multiplier in bit reversed form An output of the multiplier register is connected to a second, right shifting shifter A least significant bit of the multiplier register is connected to the arithmetic logic unit. A third register is connected to supply a multiplicand to said arithmetic logic unit. A down counter, for counting down a number corresponding to one less than the number of digits of the multiplier is connected to the arithmetic logic unit. The arithmetic logic unit is responsive to a multiply instruction to add the contents of the result register with the contents of the third register, when the least significant bit of the multiplier register is a "ONE" and to pass the contents of the result register unaltered, until the down counter completes a count The product results in the result register.

The attainment of the foregoing and related objects, advantages and features of the invention should be more readily apparent to those skilled in the art, after review of the following more detailed description of the invention, taken together with the drawings in which:

BRIEF DESCRIPTION OF THE DRAWINGS

FIG **1** is an external, plan view of an integrated circuit package incorporating a microprocessor in accordance with the invention.

FIG **2** is a block diagram of a microprocessor in accordance with the invention

FIG **3** is a block diagram of a portion of a data processing system incorporating the microprocessor of FIGS. **1** and **2**

FIG **4** is a more detailed block diagram of a portion of the microprocessor shown in FIG. **2**.

FIG. **5** is a more detailed block diagram of another portion of the microprocessor shown in FIG. **2**.

FIG. **6** is a block diagram of another portion of the data processing system shown in part in FIG. **3** and incorporating the microprocessor of FIGS. **1–2** and **4–5**.

FIGS. **7** and **8** are layout diagrams for the data processing system shown in part in FIGS. **3** and **6**.

FIG. **9** is a layout diagram of a second embodiment of a microprocessor in accordance with the invention in a data processing system on a single integrated circuit.

FIG. **10** is a more detailed block diagram of a portion of the data processing system of FIGS **7** and **8**

FIG. **11** is a timing diagram useful for understanding operation of the system portion shown in FIG. **12**.

FIG 12 is another more detailed block diagram of a further portion of the data processing system of FIGS 7 and 8

FIG. 13 is a more detailed block diagram of a portion of the microprocessor shown in FIG 2

FIG 14 is a more detailed block and schematic diagram of a portion of the system shown in FIGS 3 and 7–8.

FIG 15 is a graph useful for understanding operation of the system portion shown in FIG. 14

FIG 16 is a more detailed block diagram showing part of the system portion shown in FIG 4

FIG 17 is a more detailed block diagram of a portion of the microprocessor shown in FIG. 2

FIG 18 is a more detailed block diagram of part of the microprocessor portion shown in FIG 17

FIG 19 is a set of waveform diagrams useful for understanding operation of the part of the microprocessor portion shown in FIG. 18

FIG. 20 is a more detailed block diagram showing another part of the system portion shown in FIG 4

FIG. 21 is a more detailed block diagram showing another part of the system portion shown in FIG 4

FIGS 22 and 23 are more detailed block diagrams showing another part of the system portion shown in FIG. 4

DETAILED DESCRIPTION OF THE INVENTION

OVERVIEW

The microprocessor of this invention is desirably implemented as a 32-bit microprocessor optimized for:

HIGH EXECUTION SPEED, and

LOW SYSTEM COST.

In this embodiment, the microprocessor can be thought of as 20 MIPS for 20 dollars Important distinguishing features of the microprocessor are:

Uses low-cost commodity DYNAMIC RAMS to run 20 MIPS

4 instruction fetch per memory cycle

On-chip fast page-mode memory management

Runs fast without external cache

Requires few interfacing chips

Crams 32-bit CPU in 44 pin SOJ package

The instruction set is organized so that most operations can be specified with 8-bit instructions Two positive products of this philosophy are:

Programs are smaller,

Programs can execute much faster

The bottleneck in most computer systems is the memory bus The bus is used to fetch instructions and fetch and store data The ability to fetch four instructions in a single memory bus cycle significantly increases the bus availability to handle data

Turning now to the drawings, more particularly to FIG 1, there is shown a packaged 32-bit microprocessor **50** in a 44-pin plastic leadless chip carrier shown approximately 100 times its actual size of about 0 8 inch on a side The fact that the microprocessor **50** is provided as a 44-pin package represents a substantial departure from typical microprocessor packages, which usually have about 200 input/output (I/O) pins. The microprocessor **50** is rated at 20 million instructions per second (MIPS). Address and data lines **52**, also labelled **D0–D31**, are shared for addresses and data without speed penalty as a result of the manner in which the microprocessor **50** operates as will be explained below.

DYNAMIC RAM

In addition to the low cost 44-pin package, another unusual aspect of the high performance microprocessor **50** is

that it operates directly with dynamic random access memories (DRAMs), as shown by row address strobe (RAS) and column address strobe (CAS) I/O pins **54** The other I/O pins for the microprocessor **50** include V_{DD} pins **56**. V_{SS} pins **58**, output enable pin **60** write pin **62** clock pin **64** and reset pin **66**

All high speed computers require high speed and expensive memory to keep up The highest speed static RAM memories cost as much as ten times as much as slower dynamic RAMs This microprocessor has been optimized to use low-cost dynamic RAM in high-speed page-mode. Page-mode dynamic RAMs offer static RAM performance without the cost penalty. For example, low-cost 85 nsec. dynamic RAMs access at 25 nsec when operated in fast page-mode. Integrated fast page-mode control on the microprocessor chip simplifies system interfacing and results in a faster system.

Details of the microprocessor **50** are shown in FIG. **2** The microprocessor **50** includes a main central processing unit (CPU) **70** and a separate direct memory access (DMA) CPU **72** in a single integrated circuit making up the microprocessor **50** The main CPU **70** has a first 16 deep push down stack **74**, which has a top item register **76** and a next item register **78**, respectively connected to provide inputs to an arithmetic logic unit (ALU) **80** by lines **82** and **84** An output of the ALU **80** is connected to the top item register **76** by line **86**. The output of the top item register at **82** is also connected by line **88** to an internal data bus **90**

A loop counter **92** is connected to a decrementer **94** by lines **96** and **98** The loop counter **92** is bidirectionally connected to the internal data bus **90** by line **100**. Stack pointer **102**, return stack pointer **104**, mode register **106** and instruction register **108** are also connected to the internal data bus **90** by lines **110**, **112** **114** and **116**, respectively The internal data bus **90** is connected to memory controller **118** and to gate **120** The gate **120** provides inputs on lines **122**, **124**, and **126** to X register **128**, program counter **130** and Y register **132** of return push down stack **134** The X register **128**, program counter **130** and Y register **132** provide outputs to internal address bus **136** on lines **138** **140** and **142** The internal address bus provides inputs to the memory controller **118** and to an incrementer **144** The incrementer **144** provides inputs to the X register, program counter and Y register via lines **146**. **122** **124** and **126**. The DMA CPU **72** provides inputs to the memory controller **118** on line **148**. The memory controller **118** is connected to a RAM (not shown) by address/data bus **150** and control lines **152**

FIG. **2** shows that the microprocessor **50** has a simple architecture Prior art RISC microprocessors are substantially more complex in design. For example, the SPARC RISC microprocessor has three times the gates of the microprocessor **50** and the Intel 8960 RISC microprocessor has 20 times the gates of the microprocessor **50** The speed of this microprocessor is in substantial part due to this simplicity The architecture incorporates push down stacks and register write to achieve this simplicity.

The microprocessor **50** incorporates an I/O that has been tuned to make heavy use of resources provided on the integrated circuit chip On chip latches allow use of the same I/O circuits to handle three different things: column addressing, row addressing and data, with a slight to non-existent speed penalty. This triple bus multiplexing results in fewer buffers to expand. fewer interconnection lines, fewer I/O pins and fewer internal buffers.

The provision of on-chip DRAM control gives a performance equal to that obtained with the use of static RAMs. As a result, memory is provided at ¼ the system cost of static RAM used in most RISC systems

The microprocessor **50** fetches 4 instructions per memory cycle; the instructions are in an 8-bit format, and this is a 32-bit microprocessor. System speed is therefore 4 times the memory bus bandwidth. This ability enables the microprocessor to break the Von Neumann bottleneck of the speed of getting the next instruction. This mode of operation is possible because of the use of a push down stack and register array. The push down stack allows the use of implied addresses rather than the prior art technique of explicit addresses for two sources and a destination.

Most instructions execute in 20 nanoseconds in the microprocessor **50**. The microprocessor can therefore execute instructions at 50 peak MIPS without pipeline delays. This is a function of the small number of gates in the microprocessor **50** and the high degree of parallelism in the architecture of the microprocessor.

FIG. 3 shows how column and row addresses are multiplexed on lines D8–D14 of the microprocessor **50** for addressing DRAM **150** from I/O pins **52**. The DRAM **150** is one of eight, but only one DRAM **150** has been shown for clarity. As shown, the lines D11–D18 are respectively connected to row address inputs A0–A8 of the DRAM **150**. Additionally, lines D12–D15 are connected to the data inputs DQ1–DQ4 of the DRAM **150**. The output enable, write and column address strobe pins **54** are respectively connected to the output enable, write and column address strobe inputs of the DRAM **150** by lines **152**. The row address strobe pin **54** is connected through row address strobe decode logic **154** to the row address strobe input of the DRAM **150** by lines **156** and **158**.

D0–D7 pins **52** (FIG. 1) are idle when the microprocessor **50** is outputting multiplexed row and column addresses on D11–D18 pins **52**. The D0–D7 pins **52** can therefore simultaneously be used for I/O when right justified I/O is desired. Simultaneous addressing and I/O can therefore be carried out.

FIG. 4 shows how the microprocessor **50** is able to achieve performance equal to the use of static RAMS with DRAMs through multiple instruction fetch in a single clock cycle and instruction fetch-ahead. Instruction register **108** receives four 8-bit byte instruction words **1–4** on 32-bit internal data bus **90**. The four instruction byte **1–4** locations of the instruction register **108** are connected to multiplexer **170** by busses **172**, **174**, **176** and **178**, respectively. A microprogram counter **180** is connected to the multiplexer **170** by lines **182**. The multiplexer **170** is connected to decoder **184** by bus **186**. The decoder **184** provides internal signals to the rest of the microprocessor **50** on lines **188**.

Most significant bits **190** of each instruction byte **1–4** location are connected to a 4-input decoder **192** by lines **194**. The output of decoder **192** is connected to memory controller **118** by line **196**. Program counter **130** is connected to memory controller **118** by internal address bus **136**, and the instruction register **108** is connected to the memory controller **118** by the internal data bus **90**. Address/data bus **198** and control bus **200** are connected to the DRAMS **150** (FIG. 3).

In operation, when the most significant bits **190** of remaining instructions **1–4** are "1" in a clock cycle of the microprocessor **50**, there are no memory reference instructions in the queue. The output of decoder **192** on line **196** requests an instruction fetch ahead by memory controller **118** without interference with other accesses. While the current instructions in instruction register **108** are executing, the memory controller **118** obtains the address of the next set of four instructions from program counter **130** and obtains that set of instructions. By the time the current set of instructions has completed execution, the next set of instructions is ready for loading into the instruction register.

Details of the DMA CPU **72** are provided in FIG. 5. Internal data bus **90** is connected to memory controller **118** and to DMA instruction register **210**. The DMA instruction register **210** is connected to DMA program counter **212** by bus **214**, to transfer size counter **216** by bus **218** and to timed transfer interval counter **220** by bus **222**. The DMA instruction register **210** is also connected to DMA I/O and RAM address register **224** by line **226**. The DMA I/O and RAM address register **224** is connected to the memory controller **118** by memory cycle request line **228** and bus **230**. The DMA program counter **212** is connected to the internal address bus **136** by bus **232**. The transfer size counter **216** is connected to a DMA instruction done decrementer **234** by lines **236** and **238**. The decrementer **234** receives a control input on memory cycle acknowledge line **240**. When transfer size counter **216** has completed its count, it provides a control signal to DMA program counter **212** on line **242**. Timed transfer interval counter **220** is connected to decrementer **244** by lines **246** and **248**. The decrementer **244** receives a control input from a microprocessor system clock on line **250**.

The DMA CPU **72** controls itself and has the ability to fetch and execute instructions. It operates as a co-processor to the main CPU **70** (FIG. **2**) for time specific processing.

FIG. **6** shows how the microprocessor **50** is connected to an electrically programmable read only memory (EPROM) **260** by reconfiguring the data lines **52** so that some of the data lines **52** are input lines and some of them are output lines. Data lines **52** D0–D7 provide data to and from corresponding data terminals **262** of the EPROM **260**. Data lines **52** D9–D18 provide addresses to address terminals **264** of the EPROM **260**. Data lines **52** D19–D31 provide inputs from the microprocessor **50** to memory and I/O decode logic **266**. RAS **0/1** control line **268** provides a control signal for determining whether the memory and I/O decode logic provides a DRAM RAS output on line **270** or a column enable output for the EPROM **260** on line **272**. Column address strobe terminal **60** of the microprocessor **50** provides an output enable signal on line **274** to the corresponding terminal **276** of the EPROM **260**.

FIGS. 7 and 8 show the front and back of a one card data processing system **280** incorporating the microprocessor **50**, MSM514258-10 type DRAMs **150** totalling 2 megabytes, a Motorola 50 MegaHertz crystal oscillator clock **282**, I/O circuits **284** and a 27256 type EPROM **260**. The I/O circuits **284** include a 74HC04 type high speed hex inverter circuit **286**, an IDT39C828 type 10-bit inverting buffer circuit **288**, an IDT39C822 type 10-bit inverting register circuit **290**, and two IDT39C823 type 9-bit non-inverting register circuits **292**. The card **280** is completed with a MAX12V type DC—DC converter circuit **294**, 34-pin dual AMP type headers **296**, a coaxial female power connector **298**, and a 3-pin AMP right angle header **300**. The card **280** is a low cost, imbeddable product that can be incorporated in larger systems or used as an internal development tool.

The microprocessor **50** is a very high performance (50 MHz) RISC influenced 32-bit CPU designed to work closely with dynamic RAM. Clock for clock, the microprocessor **50** approaches the theoretical performance limits possible with a single CPU configuration. Eventually, the microprocessor **50** and any other processor is limited by the bus bandwidth and the number of bus paths. The critical conduit is between the CPU and memory.

One solution to the bus bandwidth/bus path problem is to integrate a CPU directly onto the memory chips, giving every memory a direct bus the CPU. FIG. **9** shows another microprocessor **310** that is provided integrally with 1 mega-

bit of DRAM 311 in a single integrated circuit 312 Until the present invention, this solution has not been practical, because most high performance CPUs require from 500,000 to 1 000,000 transistors and enormous die sizes just by themselves. The microprocessor 310 is equivalent to the microprocessor 50 in FIGS 1–8 The microprocessors 50 and 310 are the most transistor efficient high performance CPUs in existence requiring fewer than 50,000 transistors for dual processors 70 and 72 (FIG 2) or 314 and 316 (less memory) The very high speed of the microprocessors 50 and 310 is to a certain extent a function of the small number of active devices In essence, the less silicon gets in the way, the faster the electrons can get where they are going.

The microprocessor 310 is therefore the only CPU suitable for integration on the memory chip die 312. Some simple modifications to the basic microprocessor 50 to take advantage of the proximity to the DRAM array 311 can also increase the microprocessor 50 clock speed by 50 percent, and probably more

The microprocessor 310 core on board the DRAM die 312 provides most of the speed and functionality required for a large group of applications from automotive to peripheral control However, the integrated CPU 310/DRAM 311 concept has the potential to redefine significantly the way multiprocessor solutions can solve a spectrum of very compute intensive problems The CPU 310/DRAM 311 combination eliminates the Von Neumann bottleneck by distributing it across numerous CPU/DRAM chips 312 The microprocessor 310 is a particularly good core for multiprocessing, since it was designed with the SDI targeting array in mind, and provisions were made for efficient interprocessor communications.

Traditional multiprocessor implementations have been very expensive in addition to being unable to exploit fully the available CPU horsepower Multiprocessor systems have typically been built up from numerous board level or box level computers The result is usually an immense amount of hardware with corresponding wiring, power consumption and communications problems. By the time the systems are interconnected as much as 50 percent of the bus speed has been utilized just getting through the interfaces.

In addition, multiprocessor system software has been scarce A multiprocessor system can easily be crippled by an inadequate load-sharing algorithm in the system software, which allows one CPU to do a great deal of work and the others to be idle. Great strides have been made recently in systems software, and even UNIX V.4 may be enhanced to support multiprocessing. Several commercial products from such manufacturers as DUAL Systems and UNISOFT do a credible job on 68030 type microprocessor systems now.

The microprocessor 310 architecture eliminates most of the interface friction since up to 64 CPU 310/RAM 311 processors should be able to intercommunicate without buffers or latches. Each chip 312 has about 40 MIPS raw speed, because placing the DRAM 311 next to the CPU 310 allows the microprocessor 310 instruction cycle to be cut in half, compared to the microprocessor 50 A 64 chip array of these chips 312 is more powerful than any other existing computer Such an array fits on a 3×5 card, cost less than a FAX machine, and draw about the same power as a small television.

Dramatic changes in price/performance always reshape existing applications and almost always create new ones The introduction of microprocessors in the mid 1970s created video games, personal computers, automotive computers, electronically controlled appliances, and low cost computer peripherals

The integrated circuit 312 will find applications in all of the above areas, plus create some new ones. A common generic parallel processing algorithm handles convolution/Fast Fourier Transform (FFT)/pattern recognition. Interesting product possibilities using the integrated circuit 312 include high speed reading machines, real-time speech recognition spoken language translation, real-time robot vision, a product to identify people by their faces, and an automotive or aviation collision avoidance system.

A real time processor for enhancing high density television (HDTV) images, or compressing the HDTV information into a smaller bandwidth, would be very feasible. The load sharing in HDTV could be very straightforward Splitting up the task according to color and frame would require 6, 9 or 12 processors Practical implementation might require 4 meg RAMs integrated with the microprocessor 310.

The microprocessor 310 has the following specifications:

CONTROL LINES
4 - POWER/GROUND
1 - CLOCK
32 - DATA I/O
4 - SYSTEM CONTROL
EXTERNAL MEMORY FETCH
EXTERNAL MEMORY FETCH AUTOINCREMENT X
EXTERNAL MEMORY FETCH AUTOINCREMENT Y
EXTERNAL MEMORY WRITE
EXTERNAL MEMORY WRITE AUTOINCREMENT X
EXTERNAL MEMORY WRITE AUTOINCREMENT Y
EXTERNAL PROM FETCH
LOAD ALL X REGISTERS
LOAD ALL Y REGISTERS
LOAD ALL PC REGISTERS
EXCHANGE X AND Y
INSTRUCTION FETCH
ADD TO PC
ADD TO X
WRITE MAPPING REGISTER
READ MAPPING REGISTER
REGISTER CONFIGURATION
MICROPROCESSOR 310 CPU 316 CORE
COLUMN LATCH1 (1024 BITS) 32×32 MUX
STACK POINTER (16 BITS)
COLUMN LATCH2 (1024 BITS) 32×32 MUX
RSTACK POINTER (16 BITS)
PROGRAM COUNTER 32 BITS
X0 REGISTER 32 BITS (ACTIVATED ONLY FOR ON-CHIP ACCESSES)
Y0 REGISTER 32 BITS (ACTIVATED ONLY FOR ON-CHIP ACCESSES)
LOOP COUNTER 32 BITS
DMA CPU 314 CORE
DMA PROGRAM COUNTER 24 BITS
INSTRUCTION REGISTER 32 BITS
I/O & RAM ADDRESS REGISTER 32 BITS
TRANSFER SIZE COUNTER 12 BITS
INTERVAL COUNTER 12 BITS

To offer memory expansion for the basic chip 312, an intelligent DRAM can be produced This chip will be optimized for high speed operation with the integrated circuit 312 by having three on-chip address registers: Program Counter X Register and Y register As a result, to access the intelligent DRAM, no address is required and a total access cycle could be as short as 10 nsec. Each expansion DRAM would maintain its own copy of the three registers and would be identified by a code specifying its memory address. Incrementing and adding to the three

registers will actually take place on the memory chips A maximum of 64 intelligent DRAM peripherals would allow a large system to be created without sacrificing speed by introducing multiplexers or buffers.

There are certain differences between the microprocessor **310** and the microprocessor **50** that arise from providing the microprocessor **310** on the same die **312** with the DRAM **311** Integrating the DRAM **311** allows architectural changes in the microprocessor **310** logic to take advantage of existing on-chip DRAM **311** circuitry. Row and column design is inherent in memory architecture. The DRAMs **311** access random bits in a memory array by first selecting a row of 1024 bits, storing them into a column latch, and then selecting one of the bits as the data to be read or written.

The time required to access the data is split between the row access and the column access Selecting data already stored in a column latch is faster than selecting a random bit by at least a factor of six. The microprocessor **310** takes advantage of this high speed by creating a number of column latches and using them as caches and shift registers Selecting a new row of information may be thought of as performing a 1024-bit read or write with the resulting immense bus bandwidth

1. The microprocessor **50** treats its 32-bit instruction register **108** (see FIGS. **2** and **4**) as a cache for four 8-bit instructions. Since the DRAM **311** maintains a 1024-bit latch for the column bits, the microprocessor **310** treats the column latch as a cache for 128 8-bit instructions. Therefore, the next instruction will almost always be already present in the cache. Long loops within the cache are also possible and more useful than the 4 instruction loops in the microprocessor **50**.

2 The microprocessor **50** uses two 16×32-bit deep register arrays **74** and **134** (FIG. **2**) for the parameter stack and the return stack The microprocessor **310** creates two other 1024-bit column latches to provide the equivalent of two 32×32-bit arrays, which can be accessed twice as fast as a register array

3 The microprocessor **50** has a DMA capability which can be used for I/O to a video shift register The microprocessor **310** uses yet another 1024-bit column latch as a long video shift register to drive a CRT display directly For color displays, three on-chip shift registers could also be used. These shift registers can transfer pixels at a maximum of 100 MHz.

4. The microprocessor **50** accesses memory via an external 32-bit bus. Most of the memory **311** for the microprocessor **310** is on the same die **312** External access to more memory is made using an 8-bit bus The result is a smaller die, smaller package and lower power consumption than the microprocessor **50**.

5 The microprocessor **50** consumes about a third of its operating power charging and discharging the I/O pins and associated capacitances The DRAMs **150** (FIG. **8**) connected to the microprocessor **50** dissipate most of their power in the I/O drivers. A microprocessor **310** system will consume about one-tenth the power of a microprocessor **50** system, since having the DRAM **311** next to the processor **310** eliminates most of the external capacitances to be charged and discharged.

6 Multiprocessing means splitting a computing task between numerous processors in order to speed up the solution. The popularity of multiprocessing is limited by the expense of current individual processors as well as the limited interprocessor communications ability. The microprocessor **310** is an excellent multiprocessor candidate, since the chip **312** is a monolithic computer complete with memory, rendering it low-cost and physically compact.

The shift registers implemented with the microprocessor **310** to perform video output can also be configured as interprocessor communication links. The INMOS transputer attempted a similar strategy, but at much lower speed and without the performance benefits inherent in the microprocessor **310** column latch architecture. Serial I/O is a prerequisite for many multiprocessor topologies because of the many neighbor processors which communicate. A cube has 6 neighbors. Each neighbor communicates using these lines:

DATA IN
CLOCK IN
READY FOR DATA
DATA OUT
DATA READY?
CLOCK OUT

A special start up sequence is used to initialize the on-chip DRAM **311** in each of the processors.

The microprocessor **310** column latch architecture allows neighbor processors to deliver information directly to internal registers or even instruction caches of other chips **312** This technique is not used with existing processors, because it only improves performance in a tightly coupled DRAM system.

7. The microprocessor **50** architecture offers two types of looping structures: LOOP-IF-DONE and MICRO-LOOP The former takes an 8-bit to 24-bit operand to describe the entry point to the loop address The latter performs a loop entirely within the 4 instruction queue and the loop entry point is implied as the first instruction in the queue. Loops entirely within the queue run without external instruction fetches and execute up to three times as fast as the long loop construct. The microprocessor **310** retains both constructs with a few differences. The microprocessor **310** microloop functions in the same fashion as the microprocessor **50** operation except the queue is 1024-bits or 128 8-bit instructions long The microprocessor **310** microloop can therefore contain jumps, branches, calls and immediate operations not possible in the 4 8-bit instruction microprocessor **50** queue.

Microloops in the microprocessor **50** can only perform simple block move and compare functions The larger microprocessor **310** queue allows entire digital signal processing or floating point algorithms to loop at high speed in the queue

The microprocessor **50** offers four instructions to redirect execution:

CALL
BRANCH
BRANCH-IF-ZERO
LOOP-IF-NOT-DONE

These instructions take a variable length address operand 8, 16 or 24 bits long The microprocessor **50** next address logic treats the three operands similarly by adding or subtracting them to the current program counter For the microprocessor **310**, the 16 and 24-bit operands function in the same manner as the 16 and 24-bit operands in the microprocessor **50** The 8-bit class operands are reserved to operate entirely within the instruction queue. Next address decisions can therefore be made quickly, because only 10 bits of addresses are affected, rather than 32. There is no carry or borrow generated past the 10 bits.

8. The microprocessor **310** CPU **316** resides on an already crowded DRAM die **312** To keep chip size as small as possible, the DMA processor **72** of the microprocessor **50** has been replaced with a more traditional DMA controller **314** DMA is used with the microprocessor **310** to perform the following functions:

Video output to a CRT

Multiprocessor serial communications

8-bit parallel I/O

The DMA controller **314** can maintain both serial and parallel transfers simultaneously. The following DMA sources and destinations are supported by the microprocessor **310**:

DESCRIPTION	I/O	LINES
1 Video shift register	OUTPUT	1 to 3
2 Multiprocessor serial	BOTH	6 lines/channel
3 8-bit parallel	BOTH	8 data 4 control

The three sources use separate 1024-bit buffers and separate I/O pins. Therefore, all three may be active simultaneously without interference.

The microprocessor **310** can be implemented with either a single multiprocessor serial buffer or separate receive and sending buffers for each channel, allowing simultaneous bidirectional communications with six neighbors simultaneously.

FIGS. **10** and **11** provide details of the PROM DMA used in the microprocessor **50**. The microprocessor **50** executes faster than all but the fastest PROMs. PROMS are used in a microprocessor **50** system to store program segments and perhaps entire programs. The microprocessor **50** provides a feature on power-up to allow programs to be loaded from low-cost, slow speed PROMs into high speed DRAM for execution. The logic which performs this function is part of the DMA memory controller **118**. The operation is similar to DMA, but not identical, since four 8-bit bytes must be assembled on the microprocessor **50** chip, then written to the DRAM **150**.

The microprocessor **50** directly interfaces to DRAM **150** over a triple multiplexed data and address bus **350**, which carries RAS addresses, CAS addresses and data. The EPROM **260**, on the other hand, is read with non-multiplexed busses. The microprocessor **50** therefore has a special mode which unmultiplexes the data and address lines to read 8 bits of EPROM data. Four 8-bit bytes are read in this fashion. The multiplexed bus **350** is turned back on, and the data is written to the DRAM **150**.

When the microprocessor **50** detects a RESET condition, the processor stops the main CPU **70** and forces a mode **0** (PROM LOAD) instruction into the DMA CPU **72** instruction register. The DMA instruction directs the memory controller to read the EPROM **260** data at 8 times the normal access time for memory. Assuming a 50 MHz microprocessor **50**, this means an access time of 320 nsec. The instruction also indicates:

The selection address of the EPROM **260** to be loaded.

The number of 32-bit words to transfer,

The DRAM **150** address to transfer into.

The sequence of activities to transfer one 32-bit word from EPROM **260** to DRAM **150** are:

1. RAS goes low at **352**, latching the EPROM **260** select information from the high order address bits. The EPROM **260** is selected.
2. Twelve address bits (consisting of what is normally DRAM CAS addresses plus two byte select bits are placed on the bus **350** going to the EPROM **260** address pins. These signals will remain on the lines until the data from the EPROM **260** has been read into the microprocessor **50**. For the first byte, the byte select bits will be binary 00.
3. CAS goes low at **354**, enabling the EPROM **260** data onto the lower 8 bits of the external address/data bus **350**. NOTE: It is important to recognize that, during this part of the cycle, the lower 8 bits of the external data/address bus are functioning as inputs, but the rest of the bus is still acting as outputs.
4. The microprocessor **50** latches these eight least significant bits internally and shifts them 8 bits left to shift them to the next significant byte position.
5. Steps 2, 3 and 4 are repeated with byte address **01**.
6. Steps 2, 3 and 4 are repeated with byte address **10**.
7. Steps 2, 3 and 4 are repeated with byte address **11**.
8. CAS goes high at **356**, taking the EPROM **260** off the data bus.
9. RAS goes high at **358**, indicating the end of the EPROM **260** access.
10. RAS goes low at **360**, latching the DRAM select information from the high order address bits. At the same time, the RAS address bits are latched into the DRAM **150**. The DRAM **150** is selected.
11. CAS goes low at **362**, latching the DRAM **150** CAS addresses.
12. The microprocessor **50** places the previously latched EPROM **260** 32-bit data onto the external address/data bus **350**. W goes low at **364**, writing the 32 bits into the DRAM **150**.
13. W goes high at **366**. CAS goes high at **368**. The process continues with the next word.

FIG. **12** shows details of the microprocessor **50** memory controller **118**. In operation, bus requests stay present until they are serviced. CPU **70** requests are prioritized at **370** in the order of: 1, Parameter Stack; 2, Return Stack; 3, Data Fetch; 4, Instruction Fetch. The resulting CPU request signal and a DMA request signal are supplied as bus requests to bus control **372**, which provides a bus grant signal at **374**. Internal address bus **136** and a DMA counter **376** provide inputs to a multiplexer **378**. Either a row address or a column address are provided as an output to multiplexed address bus **380** as an output from the multiplexer **378**. The multiplexed address bus **380** and the internal data bus **90** provide address and data inputs, respectively, to multiplexer **382**. Shift register **384** supplies row address strobe (RAS) **1** and **2** control signals to multiplexer **386** and column address strobe (CAS) **1** and **2** control signals to multiplexer **388** on lines **390** and **392**. The shift register **384** also supplies output enable (OE) and write (W) signals on lines **394** and **396** and a control signal on line **398** to multiplexer **382**. The shift register **384** receives a RUN signal on line **400** to generate a memory cycle and supplies a MEMORY READY signal on line **402** when an access is complete.

STACK/REGISTER ARCHITECTURE

Most microprocessors use on-chip registers for temporary storage of variables. The on-chip registers access data faster than off-chip RAM. A few microprocessors use an on-chip push down stack for temporary storage.

A stack has the advantage of faster operation compared to on-chip registers by avoiding the necessity to select source and destination registers. (A math or logic operation always uses the top two stack items as source and the top of stack as destination.) The stack's disadvantage is that it makes some operations clumsy. Some compiler activities in particular require on-chip registers for efficiency.

As shown in FIG. 13, the microprocessor **50** provides both on-chip registers **134** and a stack **74** and reaps the benefits of both.

BENEFITS:

1. Stack math and logic is twice as fast as those available on an equivalent register only machine. Most program-

mers and optimizing compilers can take advantage of this feature

2. Sixteen registers are available for on-chip storage of local variables which can transfer to the stack for computation The accessing of variables is three to four times as fast as available on a strictly stack machine

The combined stack **74**/register **134** architecture has not been used previously due to inadequate understanding by computer designers of optimizing compilers and the mix of transfer versus math/logic instructions.

ADAPTIVE MEMORY CONTROLLER

A microprocessor must be designed to work with small or large memory configurations. As more memory loads are added to the data, address, and control lines, the switching speed of the signals slows down The microprocessor **50** multiplexes the address/data bus three ways, so timing between the phases is critical A traditional approach to the problem allocates a wide margin of time between bus phases so that systems will work with small or large numbers of memory chips connected A speed compromise of as much as 50% is required.

As shown in FIG. **14**, the microprocessor **50** uses a feedback technique to allow the processor to adjust memory bus timing to be fast with small loads and slower with large ones The OUTPUT ENABLE (OE) line **152** from the microprocessor **50** is connected to all memories **150** on the circuit board The loading on the output enable line **152** to the microprocessor **50** is directly related to the number of memories **150** connected. By monitoring how rapidly OE **152** goes high after a read, the microprocessor **50** is able to determine when the data hold time has been satisfied and place the next address on the bus.

The level of the OE line **152** is monitored by CMOS input buffer **410** which generates an internal READY signal on line **412** to the microprocessor's memory controller. Curves **414** and **416** of the FIG **15** graph show the difference in rise time likely to be encountered from a lightly to heavily loaded memory system. When the OE line **152** has reached a predetermined level to generate the READY signal driver **418** generates an OUTPUT ENABLE signal on OE line **152**

SKIP WITHIN THE INSTRUCTION CACHE

The microprocessor **50** fetches four 8-bit instructions each memory cycle and stores them in a 32-bit instruction register **108** as shown in FIG **16**. A class of "test and skip" instructions can very rapidly execute a very fast jump operation within the four instruction cache.

SKIP CONDITIONS:

Always

ACC non-zero

ACC negative

Carry flag equal logic one

Never

ACC equal zero

ACC positive

Carry flag equal logic zero

The SKIP instruction can be located in any of the four byte positions **420** in the 32-bit instruction register **108**. If the test is successful, SKIP will jump over the remaining one, two, or three 8-bit instructions in the instruction register **108** and cause the next four-instruction group to be loaded into the register **108**. As shown. the SKIP operation is implemented by resetting the 2-bit microinstruction counter **180** to zero on line **422** and simultaneously latching the next instruction group into the register **108**. Any instructions following the SKIP in the instruction register are overwritten by the new instructions and not executed.

The advantage of SKIP is that optimizing compilers and smart programmers can often use it in place of the longer conditional JUMP instruction SKIP also makes possible microloops which exit when the loop counts down or when the SKIP jumps to the next instruction group The result is very fast code.

Other machines (such as the PDP-**8** and Data General NOVA) provide the ability to skip a single instruction The microprocessor **50** provides the ability to skip up to three instructions.

MICROLOOP IN THE INSTRUCTION CACHE

The microprocessor **50** provides the MICROLOOP instruction to execute repetitively from one to three instructions residing in the instruction register **108**. The microloop instruction works in conjunction with the LOOP COUNTER **92** (FIG **2**) connected to the internal data bus **90**. To execute a microloop, the program stores a count in LOOP COUNTER **92**. MICROLOOP may be placed in the first, second, third, or last byte **420** of the instruction register **108**. If placed in the first position execution will just create a delay equal to the number stored in LOOP COUNTER **92** times the machine cycle If placed in the second. third, or last byte **420**, when the microloop instruction is executed, it will test the LOOP COUNT for zero If zero. execution will continue with the next instruction If not zero, the LOOP COUNTER **92** is decremented and the 2-bit microinstruction counter is cleared, causing the preceding instructions in the instruction register to be executed again

Microloop is useful for block move and search operations By executing a block move completely out of the instruction register **108**, the speed of the move is doubled, since all memory cycles are used by the move rather than being shared with instruction fetching. Such a hardware implementation of microloops is much faster than conventional software implementation of a comparable function.

OPTIMAL CPU CLOCK SCHEME

The designer of a high speed microprocessor must produce a product which operate over wide temperature ranges, wide voltage swings, and wide variations in semiconductor processing Temperature, voltage, and process all affect transistor propagation delays. Traditional CPU designs are done so that with the worse case of the three parameters, the circuit will function at the rated clock speed The result are designs that must be clocked a factor of two slower than their maximum theoretical performance, so they will operate properly in worse case conditions

The microprocessor **50** uses the technique shown in FIGS. **17–19** to generate the system clock and its required phases Clock circuit **430** is the familiar "ring oscillator" used to test process performance. The clock is fabricated on the same silicon chip as the rest of the microprocessor **50**.

The ring oscillator frequency is determined by the parameters of temperature, voltage and process At room temperature, the frequency will be in the neighborhood of 100 MHZ. At 70 degrees Centigrade, the speed will be 50 MHZ. The ring oscillator **430** is useful as a system clock, with its stages **431** producing phase 0-phase **3** outputs **433** shown in FIG. **19**, because its performance tracks the parameters which similarly affect all other transistors on the same silicon die. By deriving system timing from the ring oscillator **430**, CPU **70** will always execute at the maximum frequency possible, but never too fast. For example, if the processing of a particular die is not good resulting in slow transistors, the latches and gates on the microprocessor **50** will operate slower than normal. Since the microprocessor **50** ring oscillator clock **430** is made from the same transistors on the same die as the latches and gates, it too will

operate slower (oscillating at a lower frequency) providing compensation which allows the rest of the chip s logic to operate properly.

ASYNCHRONOUS/SYNCHRONOUS CPU

Most microprocessors derive all system timing from a single clock. The disadvantage is that different parts of the system can slow all operations. The microprocessor **50** provides a dual-clock scheme as shown in FIG 17, with the CPU **70** operating asynchronously to I/O interface **432** forming part of memory controller **118** (FIG 2) and the I/O interface **432** operating synchronously with the external world of memory and I/O devices. The CPU **70** executes at the fastest speed possible using the adaptive ring oscillator clock **430**. Speed may vary by a factor of four depending upon temperature, voltage, and process. The external world must be synchronized to the microprocessor **50** for operations such as video display updating and disc drive reading and writing. This synchronization is performed by the I/O interface **432**, speed of which is controlled by a conventional crystal clock **434**. The interface **432** processes requests for memory accesses from the microprocessor **50** and acknowledges the presence of I/O data. The microprocessor **50** fetches up to four instructions in a single memory cycle and can perform much useful work before requiring another memory access. By decoupling the variable speed of the CPU **70** from the fixed speed of the I/O interface **432** optimum performance can be achieved by each. Recoupling between the CPU **70** and the interface **432** is accomplished with hand shake signals on lines **436**, with data/addresses passing on bus **90, 136.**

ASYNCHRONOUS/SYNCHRONOUS CPU IMBEDDED ON A DRAM CHIP

System performance is enhanced even more when the DRAM **311** and CPU **314** (FIG 9) are located on the same die. The proximity of the transistors means that DRAM **311** and CPU **314** parameters will closely follow each other. At room temperature, not only would the CPU **314** execute at 100 MHZ, but the DRAM **311** would access fast enough to keep up. The synchronization performed by the I/O interface **432** would be for DMA and reading and writing I/O ports. In some systems (such as calculators) no I/O synchronization at all would be required and the I/O clock would be tied to the ring counter clock.

VARIABLE WIDTH OPERANDS

Many microprocessors provide variable width operands. The microprocessor **50** handles operands of 8 16, or 24 bits using the same op-code. FIG **20** shows the 32-bit instruction register **108** and the 2-bit microinstruction register **180** which selects the 8-bit instruction. Two classes of microprocessor **50** instructions can be greater than 8-bits JUMP class and IMMEDIATE. A JUMP or IMMEDIATE op-code is 8-bits, but the operand can be 8, 16, or 24 bits long. This magic is possible because operands must be right justified in the instruction register. This means that the least significant bit of the operand is always located in the least significant bit of the instruction register. The microinstruction counter **180** selects which 8-bit instruction to execute. If a JUMP or IMMEDIATE instruction is decoded, the state of the 2-bit microinstruction counter selects the required 8, 16, or 24 bit operand onto the address or data bus. The unselected 8-bit bytes are loaded with zeros by operation of decoder **440** and gates **442**. The advantage of this technique is the saving of a number of op-codes required to specify the different operand sizes in other microprocessors.

TRIPLE STACK CACHE

Computer performance is directly related to the system memory bandwidth. The faster the memories. the faster the

computer. Fast memories are expensive, so techniques have been developed to move a small amount of high-speed memory around to the memory addresses where it is needed. A large amount of slow memory is constantly updated by the fast memory, giving the appearance of a large fast memory array. A common implementation of the technique is known as a high-speed memory cache. The cache may be thought of as fast acting shock absorber smoothing out the bumps in memory access. When more memory is required than the shock can absorb, it bottoms out and slow speed memory is accessed. Most memory operations can be handled by the shock absorber itself.

The microprocessor **50** architecture has the ALU **80** (FIG 2) directly coupled to the top two stack locations **76** and **78**. The access time of the stack **74** therefore directly affects the execution speed of the processor. The microprocessor **50** stack architecture is particularly suitable to a triple cache technique, shown in FIG 21 which offers the appearance of a large stack memory operating at the speed of on-chip latches **450**. Latches **450** are the fastest form of memory device built on the chip, delivering data in as little as 3 nsec. However latches **450** require large numbers of transistors to construct. On-chip RAM **452** requires fewer transistors than latches, but is slower by a factor of five (15 nsec access). Off-chip RAM **150** is the slowest storage of all. The microprocessor **50** organizes the stack memory hierarchy as three interconnected stacks **450, 452** and **454**. The latch stack **450** is the fastest and most frequently used. The on-chip RAM stack **452** is next. The off-chip RAM stack **454** is slowest. The stack modulation determines the effective access time of the stack. If a group of stack operations never push or pull more than four consecutive items on the stack, operations will be entirely performed in the 3 nsec latch stack. When the four latches **456** are filled, the data in the bottom of the latch stack **450** is written to the top of the on-chip RAM stack **452**. When the sixteen locations **458** in the on-chip RAM stack **452** are filled, the data in the bottom of the on-chip RAM stack **452** is written to the top of the off-chip RAM stack **454**. When popping data off a full stack **450**. four pops will be performed before stack empty line **460** from the latch stack pointer **462** transfers data from the on-chip RAM stack **452**. By waiting for the latch stack **450** to empty before performing the slower on-chip RAM access, the high effective speed of the latches **456** are made available to the processor. The same approach is employed with the on-chip RAM stack **452** and the off-chip RAM stack **454**.

POLYNOMIAL GENERATION INSTRUCTION

Polynomials are useful for error correction encryption, data compression and fractal generation. A polynomial is generated by a sequence of shift and exclusive OR operations. Special chips are provided for this purpose in the prior art.

The microprocessor **50** is able to generate polynomials at high speed without external hardware by slightly modifying how the ALU **80** works. As shown in FIG **21**, a polynomial is generated by loading the 'order'' (also known as the feedback terms) into C Register **470**. The value thirty one (resulting in 32 iterations) is loaded into DOWN COUNTER **472**. A register **474** is loaded with zero. B register **476** is loaded with the starting polynomial value. When the POLY instruction executes, C register **470** is exclusively ORed with A register **474** if the least significant bit of B register **476** is a one. Otherwise, the contents of the A register **474** passes through the ALU **80** unaltered. The combination of A and B is then shifted right (divided by 2) with shifters **478** and **480**. The operation automatically repeats the specified number of iterations, and the resulting polynomial is left in A register **474**.

FAST MULTIPLY

Most microprocessors offer a 16×16 or 32×32 bit multiply instruction Multiply when performed sequentially takes one shift/add per bit, or 32 cycles for 32 bit data. The microprocessor **50** provides a high speed multiply which allows multiplication by small numbers using only a small number of cycles FIG. **23** shows the logic used to implement the high speed algorithm. To perform a multiply, the size of the multiplier less one is placed in the DOWN COUNTER 472. For a four bit multiplier, the number three would be stored in the DOWN COUNTER **472** Zero is loaded into the A register 474. The multiplier is written bit reversed into the B Register **476** For example, a bit reversed five (binary 0101) would be written into B as 1010 The multiplicand is written into the C register **470**. Executing the FAST MULT instruction will leave the result in the A Register **474**, when the count has been completed. The fast multiply instruction is important because many applications scale one number by a much smaller number. The difference in speed between multiplying a 32×32 bit and a 32×4 bit is a factor of 8. If the least significant bit of the multiplier is a "ONE", the contents of the A register **474** and the C register **470** are added. If the least significant bit of the multiplier is a "ZERO", the contents of the A register are passed through the ALU **80** unaltered. The output of the ALU **80** is shifted left by shifter **482** in each iteration. The contents of the B register **476** are shifted right by the shifter **480** in each iteration

INSTRUCTION EXECUTION PHILOSOPHY

The microprocessor **50** uses high speed D latches in most of the speed critical areas. Slower on-chip RAM is used as secondary storage

The microprocessor **50** philosophy of instruction execution is to create a hierarchy of speed as follows:

Logic and D latch transfers	1 cycle	20 nsec
Math	2 cycles	40 nsec
Fetch/store on-chip RAM	2 cycles	40 nsec
Fetch/store in current RAS page	4 cycles	80 nsec
Fetch/store with RAS cycle	11 cycles	220 nsec

With a 50 MHZ clock, many operations can be performed in 20 nsec. and almost everything else in 40 nsec

To maximize speed. certain techniques in processor design have been used. They include:

Eliminating arithmetic operations on addresses,

Fetching up to four instructions per memory cycle,

Pipelineless instruction decoding

Generating results before they are needed.

Use of three level stack caching.

PIPELINE PHILOSOPHY

Computer instructions are usually broken down into sequential pieces, for example: fetch, decode, register read, execute and store. Each piece will require a single machine cycle. In most Reduced Instruction Set Computer (RISC) chips, instruction require from three to six cycles

RISC instructions are very parallel. For example, each of 70 different instructions in the SPARC (SUN Computer's RISC chip) has five cycles. Using a technique called "pipelining", the different phases of consecutive instructions can be overlapped.

To understand pipelining, think of building five residential homes. Each home will require in sequence. a foundation. framing, plumbing and wiring, roofing, and interior finish. Assume that each activity takes one week. To build one house will take five weeks.

But what if you want to build an entire subdivision? You have only one of each work crew, but when the foundation men finish on the first house, you immediately start them on

the second one, and so on At the end of five weeks, the first home is complete, but you also have five foundations If you have kept the framing, plumbing, roofing, and interior guys all busy, from five weeks on a new house will be completed each week

This is the way a RISC chip like SPARC appears to execute an instruction in a single machine cycle In reality, a RISC chip is executing one fifth of five instructions each machine cycle And if five instructions stay in sequence, an instruction will be completed each machine cycle

The problems with a pipeline are keeping the pipe full with instructions. Each time an out of sequence instruction such as a BRANCH or CALL occurs the pipe must be refilled with the next sequence The resulting dead time to refill the pipeline can become substantial when many IF/THEN/ELSE statements or subroutines are encountered.

THE PIPELINE APPROACH

The microprocessor **50** has no pipeline as such The approach of this microprocessor to speed is to overlap instruction fetching with execution of the previously fetched instruction(s). Beyond that, over half the instructions (the most common ones) execute entirely in a single machine cycle of 20 nsec This is possible because:

1. Instruction decoding resolves in 2 5 nsec.
2. Incremented/decremented and some math values are calculated before they are needed, requiring only a latching signal to execute
3. Slower memory is hidden from high speed operations by high-speed D latches which access in 4 nsec.

The disadvantage for this microprocessor is a more complex chip design process. The advantage for the chip user is faster ultimate throughput since pipeline stalls cannot exist Pipeline synchronization with availability flag bits and other such pipeline handling is not required by this microprocessor.

For example, in some RISC machines an instruction which tests a status flag may have to wait for up to four cycles for the flag set by the previous instruction to be available to be tested Hardware and software debugging is also somewhat easier because the user doesn't have to visualize five instructions simultaneously in the pipe.

OVERLAPPING INSTRUCTION FETCH/EXECUTE

The slowest procedure the microprocessor **50** performs is to access memory. Memory is accessed when data is read or written. Memory is also read when instructions are fetched The microprocessor **50** is able to hide fetch of the next instruction behind the execution of the previously fetched instruction(s) The microprocessor **50** fetches instructions in 4-byte instruction groups. An instruction group may contain from one to four instructions The amount of time required to execute the instruction group ranges from 4 cycles for simple instructions to 64 cycles for a multiply

When a new instruction group is fetched, the microprocessor instruction decoder looks at the most significant bit of all four of the bytes. The most significant bit of an instruction determines if a memory access is required. For example, CALL FETCH, and STORE all require a memory access to execute If all four bytes have nonzero most significant bits, the microprocessor initiates the memory fetch of the next sequential 4-byte instruction group When the last instruction in the group finishes executing, the next 4-byte instruction group is ready and waiting on the data bus needing only to be latched into the instruction register If the 4-byte instruction group required four or more cycles to execute and the next sequential access was a column address strobe (CAS) cycle, the instruction fetch was completely overlapped with execution

INTERNAL ARCHITECTURE

The microprocessor **50** architecture consists of the following:

PARAMETER STACK	<--> ALU* <-->	Y REGISTER RETURN STACK
<--- 32 BITS ---> 16 DEEP		<---32 BITS---> 16 DEEP
Used for math and logic		Used for subroutine and interrupt return addresses as well as local variables
Push down stack Can overflow into off-chip RAM		Push down stack Can overflow into off-chip RAM. Can also be accessed relative to top of stack
LOOP COUNTER		(32-bits, can decrement by 1) Used by class of test and loop instructions
X REGISTER		(32-bits, can increment or decrement by 4). Used to point to RAM locations.
PROGRAM COUNTER		(32-bits, increments by 4). Points to 4-byte instruction groups in RAM
INSTRUCTION REG		(32-Bits) Holds 4-byte instruction groups while they are being decoded and executed

*Math and logic operations use the TOP item and NEXT to top Parameter Stack items as the operands. The result is pushed onto the Parameter Stack.

*Return addresses from subroutines are placed on the Return Stack. The Y REGISTER is used as a pointer to RAM locations. Since the Y REGISTER is the top item of the Return Stack nesting of indices is straightforward.

MODE - A register with mode and status bits

MODE-BITS:

Slow down memory accesses by 8 if "1" Run full speed if "0" (Provided for access to slow EPROM.)

Divide the system clock by 1023 if "1" to reduce power consumption. Run full speed if "0" (On-chip counters slow down if this bit is set.)

Enable external interrupt 1

Enable external interrupt 2

Enable external interrupt 3

Enable external interrupt 4.

Enable external interrupt 5.

Enable external interrupt 6

Enable external interrupt 7.

ON-CHIP MEMORY LOCATIONS.

MODE-BITS

DMA-POINTER

DMA-COUNTER

STACK-POINTER	Pointer into Parameter Stack.
STACK-DEPTH	Depth of on-chip Parameter Stack
RSTACK-POINTER	Pointer into Return Stack
RSTACK-DEPTH	Depth of on-chip Return Stack

ADDRESSING MODE HIGH POINTS

The data bus is 32-bits wide. All memory fetches and stores are 32-bits. Memory bus addresses are 30 bits. The least significant 2 bits are used to select one-of-four bytes in some addressing modes. The Program Counter, X Register, and Y Register are implemented as D latches with their outputs going to the memory address bus and the bus incrementer/decrementer. Incrementing one of these registers can happen quickly, because the incremented value has already rippled through the inc/dec logic and need only be clocked into the latch. Branches and Calls are made to 32-bit word-boundaries.

INSTRUCTION SET

32-BIT INSTRUCTION FORMAT

The thirty two bit instructions are CALL, BRANCH BRANCH-IF-ZERO, and LOOP-IF-NOT-DONE. These instructions require the calculation of an effective address. In many computers, the effective address is calculated by adding or subtracting an operand with the current Program Counter. This math operation requires from four to seven machine cycles to perform and can definitely bog down machine execution. The microprocessor's strategy is to perform the required math operation at assembly or linking time and do a much simpler "Increment to next page" or "Decrement to previous page" operation at run time. As a result, the microprocessor branches execute in a single cycle.

24-BIT OPERAND FORM:

Byte 1 Byte 2 Byte 3 Byte 4

WWWWWW XX - YYYYYYYY - YYYYYYYY - YYYYYYYY

With a 24-bit operand, the current page is considered to be defined by the most significant 6 bits of the Program Counter

16-BIT OPERAND FORM:

QQQQQQQQ - WWWWWW XX - YYYYYYYY - YYYYYYYY

With a 16-bit operand, the current page is considered to be defined by the most significant 14 bits of the Program Counter.

8-BIT OPERAND FORM:

QQQQQQQQ - QQQQQQQQ - WWWWWW XX - YYYYYYYY

With an 8-bit operand, the current page is considered to be defined by the most significant 22 bits of the Program Counter

QQQQQQQQ - Any 8-bit instruction

WWWWWW - Instruction op-code.

XX - Select how the address bits will be used:

00 - Make all high-order bits zero (Page zero addressing)

01 - Increment the high-order bits. (Use next page)

10 - Decrement the high-order bits (Use previous page)

11 - Leave the high-order bits unchanged (Use current page)

YYYYYYYY - The address operand field. This field is always shifted left two bits (to generate a word rather than byte address) and loaded into the Program Counter. The microprocessor instruction decoder figures out the width of the operand field by the location of the instruction op-code in the four bytes

The compiler or assembler will normally use the shortest operand required to reach the desired address so that the leading bytes can be used to hold other instructions. The effective address is calculated by combining:

The current Program Counter,

The 8, 16, or 24 bit address operand in the instruction

Using one of the four allowed addressing modes

EXAMPLES OF EFFECTIVE ADDRESS CALCULATION

Example 1:

Byte 1	Byte 2	Byte 3	Byte 4
QQQQQQQQ	QQQQQQQQ	00000011	10011000

The 'QQQQQQQQs' in Byte **1** and **2** indicate space in the 4-byte memory fetch which could be hold two other instructions to be executed prior to the CALL instruction. Byte **3** indicates a CALL instruction (six zeros) in the current page (indicated by the 11 bits). Byte **4** indicates that the hexadecimal number 98 will be forced into the Program Counter bits **2** through **10** (Remember, a CALL or BRANCH always goes to a word boundary so the two least

significant bits are always set to zero) The effect of this instruction would be to CALL a subroutine at WORD location HEX 98 in the current page. The most significant 22 bits of the Program Counter define the current page and will be unchanged.

Example 2:

Byte 1	Byte 2	Byte 3	Byte 4
000001 01	00000001	00000000	00000000

If we assume that the Program Counter was HEX 0000 0156 which is binary:

00000000 00000000 00000001 01010110 = OLD PROGRAM COUNTER

Byte 1 indicates a BRANCH instruction op code (000001) and "01" indicates select the next page Byte 2,3, and 4 are the address operand These 24-bits will be shifted to the left two places to define a WORD address. HEX 0156 shifted left two places is HEX 0558. Since this is a 24-bit operand instruction, the most significant 6 bits of the Program Counter define the current page These six bits will be incremented to select the next page Executing this instruction will cause the Program Counter to be loaded with HEX 0400 0558 which is binary:

00000100 00000000 00000101 01011000 = NEW PROGRAM COUNTER.

INSTRUCTIONS

CALL-LONG

0000 00XX - YYYYYYYY - YYYYYYYY - YYYYYYYY

Load the Program Counter with the effective WORD address specified Push the current PC contents onto the RETURN STACK.

OTHER EFFECTS: CARRY or modes, no effect May cause Return Stack to force an external memory cycle if on-chip Return Stack is full

BRANCH

0000 01XX - YYYYYYYY - YYYYYYYY - YYYYYYYY

Load the Program Counter with the effective WORD address specified.

OTHER EFFECTS: NONE

BRANCH-IF-ZERO

0000 10XX - YYYYYYYY - YYYYYYYY - YYYYYYYY

Test the TOP value on the Parameter Stack If the value is equal to zero, load the Program Counter with the effective WORD address specified. If the TOP value is not equal to zero, increment the Program Counter and fetch and execute the next instruction.

OTHER EFFECTS: NONE

LOOP-IF-NOT-DONE

0000 11YY - (XXXX XXXX) - (XXXX XXXX) - (XXXX XXXX)

If the LOOP COUNTER is not zero, load the Program Counter with the effective WORD address specified If the LOOP COUNTER is zero, decrement the LOOP COUNTER, increment the Program Counter and fetch and execute the next instruction.

OTHER EFFECTS: NONE

8-BIT INSTRUCTIONS PHILOSOPHY

Most of the work in the microprocessor **50** is done by the 8-bit instructions Eight bit instructions are possible with the microprocessor because of the extensive use of implied stack addressing Many 32-bit architectures use 8-bits to specify the operation to perform but use an additional 24-bits to specify two sources and a destination

For math and logic operations, the microprocessor **50** exploits the inherent advantage of a stack by designating the source operand(s) as the top stack item and the next stack item The math or logic operation is performed, the operands are popped from the stack and the result is pushed back on the stack The result is a very efficient utilization of instruction bits as well as registers A comparable situation exists between Hewlett Packard calculators (which use a stack) and Texas Instrument calculators which don't The identical operation on an HP will require one half to one third the keystrokes of the TI.

The availability of 8-bit instructions also allows another architectural innovation the fetching of four instructions in a single 32-bit memory cycle. The advantages of fetching multiple instructions are:

Increased execution speed even with slow memories.

Similar performance to the Harvard (separate data and instruction busses) without the expense,

Opportunities to optimize groups of instructions,

The capability to perform loops within this mini-cache.

The microloops inside the four instruction group are effective for searches and block moves.

SKIP INSTRUCTIONS

The microprocessor **50** fetches instructions in 32-bit chunks called 4-byte instruction groups. These four bytes may contain four 8-bit instructions or some mix of 8-bit and 16 or 24-bit instructions SKIP instructions in the microprocessor skip any remaining instructions in a 4-byte instruction group and cause a memory fetch to get the next 4-byte instruction group. Conditional SKIPs when combined with 3-byte BRANCHES will create conditional BRANCHES. SKIPs may also be used in situations when no use can be made of the remaining bytes in a 4-instruction group. A SKIP executes in a single cycle, whereas a group of three NOPs would take three cycles.

SKIP-ALWAYS -	skip any remaining instructions in this 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group.
SKIP-IF-ZERO -	If the TOP item of the Parameter Stack is zero, skip any remaining instructions in the 4-byte instruction group Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group If the TOP item is not zero, execute the next sequential instruction.
SKIP-IF-POSITIVE -	If the TOP item of the Parameter Stack has a the most significant bit (the sign bit) equal to "0", skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group If the TOP item is not "0", execute the next sequential instruction

-continued

SKIP-IF-NO-CARRY -	If the CARRY flag from a SHIFT or arithmetic operation is not equal to "1", skip any remaining instructions in the 4-byte instruction group Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the CARRY is equal to "1" execute the next sequential instruction
SKIP-NEVER (NOP)	Execute the next sequential instruction (Delay one machine cycle).
SKIP-IF-NOT-ZERO -	If the TOP item on the Parameter Stack is not equal to "0", skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group If the TOP item is equal 0 , execute the next sequential instruction
SKIP-IF-NEGATIVE -	If the TOP item on the Parameter Stack has its most significant bit (sign bit) set to "1" skip any remaining instructions in the 4-byte instruction group Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the TOP item has its most significant bit set to "0", execute the next sequential instruction.
SKIP-IF-CARRY -	If the CARRY flag is set to 1 as a result of SHIFT or arithmetic operation, skip any remaining instructions in the 4-byte instruction group. Increment the most significant 30-bits of the Program Counter and proceed to fetch the next 4-byte instruction group. If the CARRY flag is "0", execute the next sequential instruction.

MICROLOOPS

Microloops are a unique feature of the microprocessor architecture which allows controlled looping within a 4-byte instruction group. A microloop instruction tests the LOOP COUNTER for "0" and may perform an additional test If the LOOP COUNTER is not "0" and the test is met, instruction execution continues with the first instruction in the 4-byte instruction group and the LOOP COUNTER is decremented. A microloop instruction will usually be the last byte in a 4-byte instruction group, but it can be any byte If the LOOP COUNTER is "0" or the test is not met, instruction execution continues with the next instruction. If the microloop is the last byte in the 4-byte instruction group, the most significant 30-bits of the Program Counter are incremented and the next 4-byte instruction group is fetched from memory On a termination of the loop on LOOP COUNTER equal to "0", the LOOP COUNTER will remain at "0". Microloops allow short iterative work such as moves and searches to be performed without slowing down to fetch instructions from memory

EXAMPLE:

Byte 1	Byte 2
FETCH-VIA-X-AUTOINCREMENT	STORE-VIA-Y-AUTO-INCREMENT
Byte 3	Byte 4
ULOOP-UNTIL-DONE	QQQQQQQQ

This example will perform a block move To initiate the transfer, X will be loaded with the starting address of the source Y will be loaded with the starting address of the destination. The LOOP COUNTER will be loaded with the number of 32-bit words to move The microloop will FETCH and STORE and count down the LOOP COUNTER until it reaches zero QQQQQQQQ indicates any instruction can follow.

MICROLOOP INSTRUCTIONS

ULOOP-UNTIL-DONE—If the LOOP COUNTER is not "0", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER If the LOOP COUNTER is "0", continue execution with the next instruction

ULOOP-IF-ZERO—If the LOOP COUNTER is not "0" and the TOP item on the Parameter Stack is "0", continue execution with the first instruction in the 4-byte instruction group Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the TOP item is "1", continue execution with the next instruction.

ULOOP-IF-POSITIVE—If the LOOP COUNTER is not "0" and the most significant bit (sign bit) is "0", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER If the LOOP COUNTER is "0" or the TOP item is "1", continue execution with the next instruction.

ULOOP-IF-NOT-CARRY-CLEAR—If the LOOP COUNTER is not "0" and the floating point exponents found in TOP and NEXT are not aligned continue execution with the first instruction in the 4-byte instruction group Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the exponents are aligned, continue execution with the next instruction This instruction is specifically designed for combination with special SHIFT instructions to align two floating point numbers.

ULOOP-NEVER—(DECREMENT-LOOP-COUNTER) Decrement the LOOP COUNTER. Continue execution with the next instruction.

ULOOP-IF-NOT-ZERO—If the LOOP COUNTER is not "0" and the TOP item of the Parameter Stack is "0", continue execution with the first instruction in the 4-byte instruction group Decrement the LOOP COUNTER. If the LOOP COUNTER is "0" or the TOP item is "1", continue execution with the next instruction

ULOOP-IF-NEGATIVE—If the LOOP COUNTER is not "0" and the most significant bit (sign bit) of the TOP item of the Parameter Stack is "1", continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER If the LOOP COUNTER is "0" or the most significant bit of the Parameter Stack is "0", continue execution with the next instruction.

ULOOP-IF-CARRY-SET—If the LOOP COUNTER is not "0" and the exponents of the floating point numbers found in TOP and NEXT are not aligned, continue execution with the first instruction in the 4-byte instruction group. Decrement the LOOP COUNTER. If the LOOP COUNTER is "0 or the exponents are aligned, continue execution with the next instruction.

RETURN FROM SUBROUTINE OR INTERRUPT

Subroutine calls and interrupt acknowledgements cause a redirection of normal program execution. In both cases, the current Program Counter is pushed onto the Return Stack, so the microprocessor can return to its place in the program after executing the subroutine or interrupt service routine

NOTE: When a CALL to subroutine or interrupt is acknowledged the Program Counter has already been incremented and is pointing to the 4-byte instruction group following the 4-byte group currently being executed The instruction decoding logic allows the microprocessor to

perform a test and execute a return conditional on the outcome of the test in a single cycle. A RETURN pops an address from the Return Stack and stores it to the Program Counter

RETURN INSTRUCTIONS	
RETURN-ALWAYS -	Pop the top item from the Return Stack and transfer it to the Program Counter.
RETURN-IF-ZERO -	If the TOP item on the Parameter Stack is "0", pop the top item from the Return Stack and transfer it to the Program Counter Otherwise execute the next instruction.
RETURN-IF-POSITIVE -	If the most significant bit (sign bit) of the TOP item on the Parameter Stack is a "0", pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction
RETURN-IF-CARRY-CLEAR -	If the exponents of the floating point numbers found in TOP and NEXT are not aligned pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.
RETURN-NEVER - (NOP)	Execute the next instruction
RETURN-IF-NOT-ZERO -	If the TOP item on the Parameter Stack is not "0", pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction
RETURN-IF-NEGATIVE -	If the most significant bit (sign bit) of the TOP item on the Parameter Stack is a "1", pop the top item from the Return Stack and transfer it to the Program Counter. Otherwise execute the next instruction.
RETURN-IF-CARRY-SET -	If the exponents of the floating point numbers found in TOP and NEXT are aligned, pop the top item from the Return Stack and transfer it to the Program. Counter. Otherwise execute the next instruction.

HANDLING MEMORY FROM DYNAMIC RAM

The microprocessor **50**, like any RISC type architecture, is optimized to handle as many operations as possible on-chip for maximum speed. External memory operations take from 80 nsec. to 220 nsec. compared with on-chip memory speeds of from 4 nsec to 30 nsec. There are times when external memory must be accessed.

External memory is accessed using three registers:

X-REGISTER—A 30-bit memory pointer which can be used for memory access and simultaneously incremented or decremented.

Y-REGISTER—A 30-bit memory pointer which can be used for memory access and simultaneously incremented or decremented

PROGRAM-COUNTER—A 30-bit memory pointer normally used to point to 4-byte instruction groups External memory may be accessed at addresses relative to the PC. The operands are sometimes called "Immediate" or "Literal" in other computers When used as memory pointer, the PC is also incremented after each operation.

MEMORY LOAD & STORE INSTRUCTIONS

FETCH-VIA-X—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack X is unchanged

FETCH-VIA-Y—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack. Y is unchanged

FETCH-VIA-X-AUTOINCREMENT—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack. After fetching increment the most significant 30 bits of X to point to the next 32-bit word address.

FETCH-VIA-Y-AUTOINCREMENT—Fetch the 32-bit memory content pointed to by Y and push it onto the Parameter Stack After fetching, increment the most significant 30 bits of Y to point to the next 32-bit word address.

FETCH-VIA-X-AUTODECREMENT—Fetch the 32-bit memory content pointed to by X and push it onto the Parameter Stack After fetching, decrement the most significant 30 bits of X to point to the previous 32-bit word address.

FETCH-VIA-Y-AUTODECREMENT—Fetch the 32-bit memory content pointed to by Y and push it onto the Parameter Stack. After fetching, decrement the most significant 30 bits of Y to point to the previous 32-bit word address.

STORE-VIA-X—Pop the top item of the Parameter Stack and store it in the memory location pointed to by X. X is unchanged.

STORE-VIA-Y—Pop the top item of the Parameter Stack and store it in the memory location pointed to by Y. Y is unchanged.

STORE-VIA-X-AUTOINCREMENT—Pop the top item of the Parameter Stack and store it in the memory location pointed to by X. After storing, increment the most significant 30 bits of X to point to the next 32-bit word address.

STORE-VIA-Y-AUTOINCREMENT—Pop the top item of the Parameter Stack and store it in the memory location pointed to by Y After storing, increment the most significant 30 bits of Y to point to the next 32-bit word address.

STORE-VIA-X-AUTODECREMENT—Pop the top item of the Parameter Stack and store it in the memory location pointed to by X After storing, decrement the most significant 30 bits of X to point to the previous 32-bit word address.

STORE-VIA-Y-AUTODECREMENT—Pop the top item of the Parameter Stack and store it in the memory location pointed to by Y After storing, decrement the most significant 30 bits of Y to point to the previous 32-bit word address.

FETCH-VIA-PC—Fetch the 32-bit memory content pointed to by the Program Counter and push it onto the Parameter Stack. After fetching, increment the most significant 30 bits of the Program Counter to point to the next 32-bit word address.

*NOTE When this instruction executes, the PC is pointing to the memory location following the instruction. The effect is of loading a 32-bit immediate operand This is an 8-bit instruction and therefore will be combined with other 8-bit instructions in a 4-byte instruction fetch It is possible to have from one to four FETCH-VIA-PC instructions in a 4-byte instruction fetch The PC incre-

ments after each execution of FETCH-VIA-PC, so it is possible to push four immediate operands on the stack. The four operands would be the found in the four memory locations following the instruction.

BYTE-FETCH-VIA-X—Fetch the 32-bit memory content pointed to by the most significant 30 bits of X. Using the two least significant bits of X, select one of four bytes from the 32-bit memory fetch, right justify the byte in a 32-bit field and push the selected byte preceded by leading zeros onto the Parameter Stack.

BYTE-STORE-VIA-X—Fetch the 32-bit memory content pointed to by the most significant 30 bits of X. Pop the TOP item from the Parameter Stack. Using the two least significant bits of X place the least significant byte into the 32-bit memory data and write the 32-bit entity back to the location pointed to by the most significant 30 bits of X.

OTHER EFFECTS OF MEMORY ACCESS INSTRUCTIONS:

Any FETCH instruction will push a value on the Parameter Stack 74. If the on-chip stack is full, the stack will overflow into off-chip memory stack resulting in an additional memory cycle. Any STORE instruction will pop a value from the Parameter Stack 74. If the on-chip stack is empty, a memory cycle will be generated to fetch a value from off-chip memory stack.

HANDLING ON-CHIP VARIABLES

High-level languages often allow the creation of LOCAL VARIABLES. These variables are used by a particular procedure and discarded. In cases of nested procedures, layers of these variables must be maintained. On-chip storage is up to five times faster than off-chip RAM, so a means of keeping local variables on-chip can make operations run faster. The microprocessor **50** provides the capability for both on-chip storage of local variables and nesting of multiple levels of variables through the Return Stack.

The Return Stack **134** is implemented as 16 on-chip RAM locations. The most common use for the Return Stack **134** is storage of return addresses from subroutines and interrupt calls. The microprocessor allows these 16 locations to also be used as addressable registers. The 16 locations may be read and written by two instructions which indicate a Return Stack relative address from 0–15. When high-level procedures are nested, the current procedure variables push the previous procedure variables further down the Return Stack **134**. Eventually, the Return Stack will automatically overflow into off-chip RAM.

ON-CHIP VARIABLE INSTRUCTIONS

READ-LOCAL-VARIABLE XXXX—Read the XXXXth location relative to the top of the Return Stack. (XXXX is a binary number from 0000–1111). Push the item read onto the Parameter Stack.

OTHER EFFECTS: If the Parameter Stack is full, the push operation will cause a memory cycle to be generated as one item of the stack is automatically stored to external RAM. The logic which selects the location performs a modulo 16 subtraction. If four local variables have been pushed onto the Return Stack, and an instruction attempts to READ the fifth item, unknown data will be returned.

WRITE-LOCAL-VARIABLE XXXX—Pop the TOP item of the Parameter Stack and write it into the XXXXth location relative to the top of the Return Stack. (XXXX is a binary number from 0000–1111.)

OTHER EFFECTS: If the Parameter Stack is empty, the pop operation will cause a memory cycle to be generated to fetch the Parameter Stack item from external RAM. The logic which selects the location performs a modulo 16 subtraction. If four local variables have been pushed onto the Return Stack, and an instruction attempts to WRITE to the fifth item it is possible to clobber return addresses or wreak other havoc.

REGISTER AND FLIP-FLOP TRANSFER AND PUSH INSTRUCTIONS

DROP—Pop the TOP item from the Parameter Stack and discard it.

SWAP—Exchange the data in the TOP Parameter Stack location with the data in the NEXT Parameter Stack location.

DUP—Duplicate the TOP item on the Parameter Stack and push it onto the Parameter Stack.

PUSH-LOOP-COUNTER—Push the value in LOOP COUNTER onto the Parameter Stack.

POP-RSTACK-PUSH-TO-STACK—Pop the top item from the Return Stack and push it onto the Parameter Stack.

PUSH-X-REG—Push the value in the X Register onto the Parameter Stack.

PUSH-STACK-POINTER—Push the value of the Parameter Stack pointer onto the Parameter Stack.

PUSH-RSTACK-POINTER—Push the value of the Return Stack pointer onto the Return Stack.

PUSH-MODE-BITS—Push the value of the MODE REGISTER onto the Parameter Stack.

PUSH-INPUT—Read the 10 dedicated input bits and push the value (right justified and padded with leading zeros) onto the Parameter Stack.

SET-LOOP-COUNTER—Pop the TOP value from the Parameter Stack and store it into LOOP COUNTER.

POP-STACK-PUSH-TO-RSTACK—Pop the TOP item from the Parameter Stack and push it onto the Return Stack.

SET-X-REG—Pop the TOP item from the Parameter Stack and store it into the X Register.

SET-STACK-POINTER—Pop the TOP item from the Parameter Stack and store it into the Stack Pointer.

SET-RSTACK-POINTER—Pop the TOP item from the Parameter Stack and store it into the Return Stack Pointer.

SET-MODE-BITS—Pop the TOP value from the Parameter Stack and store it into the MODE BITS.

SET-OUTPUT—Pop the TOP item from the Parameter Stack and output it to the 10 dedicated output bits.

OTHER EFFECTS: Instructions which push or pop the Parameter Stack or Return Stack may cause a memory cycle as the stacks overflow back and forth between on-chip and off-chip memory.

LOADING A SHORT LITERAL

A special case of register transfer instruction is used to push an 8-bit literal onto the Parameter Stack. This instruction requires that the 8-bits to be pushed reside in the last byte of a 4-byte instruction group. The instruction op-code loading the literal may reside in ANY of the other three bytes in the instruction group.

EXAMPLE:

BYTE 1	BYTE 2	BYTE 3
LOAD-SHORT-LITERAL	QQQQQQQQ	QQQQQQQQ
BYTE 4		
00001111		

In this example, QQQQQQQQ indicates any other 8-bit instruction. When Byte 1 is executed, binary 00001111 (HEX 0f) from Byte 4 will be pushed (right justified and padded by leading zeros) onto the Parameter Stack. Then the instructions in Byte 2 and Byte 3 will execute. The microprocessor instruction decoder knows not to execute Byte 4. It is possible to push three identical 8-bit values as follows:

BYTE 1	BYTE 2
LOAD-SHORT-LITERAL	LOAD-SHORT-LITERAL
BYTE 3	BYTE 4
LOAD-SHORT-LITERAL	00001111
SHORT-LITERAL-INSTRUCTION	
LOAD-SHORT-LITERAL -	Push the 8-bit value found in Byte 4 of the current 4-byte instruction group onto the Parameter Stack

LOGIC INSTRUCTIONS

Logical and math operations used the stack for the source of one or two operands and as the destination for results The stack organization is a particularly convenient arrangement for evaluating expressions. TOP indicates the top value on the Parameter Stack 74. NEXT indicates the next to top value on the Parameter Stack 74

AND—Pop TOP and NEXT from the Parameter Stack perform the logical AND operation on these two operands, and push the result onto the Parameter Stack

OR—Pop TOP and NEXT from the Parameter Stack, perform the logical OR operation on these two operands. and push the result onto the Parameter Stack

XOR—Pop TOP and NEXT from the Parameter Stack, perform the logical exclusive OR on these two operands, and push the result onto the Parameter Stack.

BIT-CLEAR—Pop TOP and NEXT from the Parameter Stack, toggle all bits in NEXT, perform the logical AND operation on TOP and push the result onto the Parameter Stack (Another way of understanding this instruction is thinking of it as clearing all bits in TOP that are set in NEXT.)

MATH INSTRUCTIONS

Math instruction pop the TOP item and NEXT to top item of the Parameter Stack 74 to use as the operands. The results are pushed back on the Parameter Stack The CARRY flag is used to latch the "33rd bit' of the ALU result

ADD—Pop the TOP item and NEXT to top item from the Parameter Stack, add the values together and push the result back on the Parameter Stack The CARRY flag may be changed.

ADD-WITH-CARRY—Pop the TOP item and the NEXT to top item from the Parameter Stack. add the values together If the CARRY flag is '1' increment the result. Push the ultimate result back on the Parameter Stack The CARRY flag may be changed

ADD-X—Pop the TOP item from the Parameter Stack and read the third item from the top of the Parameter Stack. Add the values together and push the result back on the Parameter Stack. The CARRY flag may be changed

SUB—Pop the TOP item and NEXT to top item from the Parameter Stack, Subtract NEXT from TOP and push the result back on the Parameter Stack The CARRY flag may be changed.

SUB-WITH-CARRY—Pop the TOP item and NEXT to top item from the Parameter Stack Subtract NEXT from TOP. If the CARRY flag is "1" increment the result Push the ultimate result back on the Parameter Stack. The CARRY flag may be changed.

SUB-X—

SIGNED-MULT-STEP—

UNSIGNED-MULT-STEP—

SIGNED-FAST-MULT—

FAST-MULT-STEP—

UNSIGNED-DIV-STEP—

GENERATE-POLYNOMIAL

ROUND—

COMPARE—Pop the TOP item and NEXT to top item from the Parameter Stack. Subtract NEXT from TOP If the result has the most significant bit equal to "0' (the result is positive), push the result onto the Parameter Stack. If the result has the most significant bit equal to "1" (the result is negative), push the old value of TOP onto the Parameter Stack The CARRY flag may be affected

SHIFT/ROTATE

SHIFT-LEFT—Shift the TOP Parameter Stack item left one bit. The CARRY flag is shifted into the least significant bit of TOP.

SHIFT-RIGHT—Shift the TOP Parameter Stack item right one bit. The least significant bit of TOP is shifted into the CARRY flag Zero is shifted into the most significant bit of TOP

DOUBLE-SHIFT-LEFT—Treating the TOP item of the Parameter Stack as the most significant word of a 64-bit number and the NEXT stack item as the least significant word, shift the combined 64-bit entity left one bit. The CARRY flag is shifted into the least significant bit of NEXT.

DOUBLE-SHIFT-RIGHT—Treating the TOP item of the Parameter Stack as the most significant word of a 64-bit number and the NEXT stack item as the least significant word shift the combined 64-bit entity right one bit. The least significant bit of NEXT is shifted into the CARRY flag. Zero is shifted into the most significant bit of TOP.

OTHER INSTRUCTIONS

FLUSH-STACK—Empty all on-chip Parameter Stack locations into off-chip RAM (This instruction is useful for multitasking applications). This instruction accesses a counter which holds the depth of the on-chip stack and can require from none to 16 external memory cycles.

FLUSH-RSTACK—Empty all on-chip Return Stack locations into off-chip RAM (This instruction is useful for multitasking applications) This instruction accesses a counter which holds the depth of the on-chip Return Stack and can require from none to 16 external memory cycles.

It should further be apparent to those skilled in the art that various changes in form and details of the invention as shown and described may be made It is intended that such changes be included within the spirit and scope of the claims appended hereto

What is claimed is:

1. A microprocessor, which comprises a main central processing unit and a separate direct memory access central processing unit in a single integrated circuit comprising said microprocessor, said main central processing unit having an arithmetic logic unit, a first push down stack with a top item register and a next item register, connected to provide inputs to said arithmetic logic unit, an output of said arithmetic logic unit being connected to said top item register, said top item register also being connected to provide inputs to an internal data bus, said internal data bus being bidirectionally connected to a loop counter, said loop counter being connected to a decrementer, said internal data bus being bidirectionally connected to a stack pointer, return stack pointer., mode register and instruction register, said internal data bus being connected to a memory controller, to a Y register of a return push down stack, an X register and a program counter, said Y register X register and program counter providing outputs to an internal address bus, said internal address bus providing inputs to said memory controller and to an incrementer, said incrementer being connected to said internal data bus, said direct memory access central processing unit providing inputs to said memory controller, said memory controller having an address/data bus and a plurality of control lines for connection to a random access memory.

2. The microprocessor of claim 1 in which said memory controller includes a multiplexing means between said central processing unit and said address/data bus, said multiplexing means being connected and configured to provide row addresses, column addresses and data on said address/data bus.

3. The microprocessor of claim 1 in which said memory controller includes means for fetching instructions for said central processing unit on said address/data bus, said means for fetching instructions being configured to fetch multiple sequential instructions in a single memory cycle.

4. The microprocessor of claim 3 additionally comprising means connected to said means for fetching instructions for determining if multiple instructions fetched by said means for fetching instructions require a memory access, said means for fetching instructions fetching additional multiple instructions if the multiple instructions do not require a memory access.

5. The microprocessor of claim 3 in which said microprocessor and a dynamic random access memory are contained in a single integrated circuit and said means for fetching instructions includes a column latch for receiving the multiple instructions.

6. The microprocessor of claim 1 in which said microprocessor includes a sensing circuit and a driver circuit, and an output enable line for connection between the random access memory, said sensing circuit and said driver circuit, said sensing circuit being configured to provide a ready signal when said output enable line reaches a predetermined electrical level, said microprocessor being configured so that said driver circuit provides an enabling signal on said output enable line responsive to the ready signal.

7. The microprocessor of claim 1 additionally comprising a ring oscillator variable speed system clock connected to said main central processing unit, said main central processing unit and said ring oscillator variable speed system clock being provided in a single integrated circuit.

8. The microprocessor of claim 7 in which said memory controller includes an input/output interface connected to exchange coupling control signals, addresses and data with said main central processing unit, said microprocessor additionally including a second clock independent of said ring oscillator variable speed system clock connected to said input/output interface.

9. The microprocessor of claim 1 in which said first push down stack has a first plurality of stack elements configured as latches, a second plurality of stack elements configured as a random access memory, said first and second plurality of stack elements and said central processing unit being provided in a single integrated circuit, and a third plurality of stack elements configured as a random access memory external to said single integrated circuit.

10. The microprocessor of claim 9 additionally comprising a first pointer connected to said first plurality of stack elements, a second pointer connected to said second plurality of stack elements, and a third pointer connected to said third plurality of stack elements, said central processing unit being connected to pop items from said first plurality of stack elements, said first stack pointer being connected to said second stack pointer to pop a first plurality of items from said second plurality of stack elements when said first plurality of stack elements are empty from successive pop operations by said central processing unit, said second stack pointer being connected to said third stack pointer to pop a second plurality of items from said third plurality of stack elements when said second plurality of stack elements are empty from successive pop operations by said central processing unit.

* * * * *

EXHIBIT D

TO THE DECLARATION OF JEFFREY M. FISHER ISO DEFENDANTS' REPLY ISO MOTION TO DISMISS OR, IN THE ALTERNATIVE, TO TRANSFER VENUE

CLOSED, PATENT

U.S. District Court [LIVE]
Eastern District of TEXAS (Tyler)
CIVIL DOCKET FOR CASE #: 6:05-cv-00013-LED

Mosaid Technologies Inc v. Hynix Semiconductor Inc et al
Assigned to: Judge Leonard Davis
Cause: 35:271 Patent Infringement

Date Filed: 01/18/2005
Date Terminated: 02/28/2005
Jury Demand: Plaintiff
Nature of Suit: 830 Patent
Jurisdiction: Federal Question

Plaintiff

Mosaid Technologies Inc

represented by **Kenneth Robert Adamo**
Jones Day - Dallas
2727 North Harwood Street
Dallas, Tx 75201
214-969-4856
Email: kradamo@jonesday.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Michael Edwin Jones
Potter Minton PC
110 N College
Suite 500
PO Box 359
Tyler, TX 75710-0359
903/597/8311
Fax: 9035930846
Email: mikejones@potterminton.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Cindy Marie Allen
Potter Minton
110 N. College
Suite 500
Tyler, TX 75710
903/597-8311
Fax: 903593046
Email: cmallen@potterminton.com

V.

Defendant

Hynix Semiconductor Inc

Defendant

Hynix Semiconductor America Inc

Defendant

Hynix Semiconductor Manufacturing America, Inc

Date Filed	#	Docket Text
01/18/2005	1	COMPLAINT against Hynix Semiconductor Inc, Hynix Semiconductor America Inc, Hynix Semiconductor Manufacturing America, Inc , filed by Mosaid Technologies Inc. (Attachments: # 1 Civil Cover Sheet)(mll,) (Entered: 01/19/2005)
01/18/2005	2	DEMAND for Trial by Jury by Mosaid Technologies Inc. (mll,) (Entered: 01/19/2005)
01/18/2005	3	NOTICE OF CASE ASSIGNMENT cc:pltf 1-19-05 (mll,) (Entered: 01/19/2005)
01/18/2005		Filing fee: $ 150.00, receipt number 629705 (mll,) (Entered: 01/19/2005)
01/18/2005	4	Form mailed to Commissioner of Patents and Trademarks. (mll,) (Entered: 01/19/2005)
01/20/2005	5	ORDER OF RECUSAL. Judge William M. Steger recused. Case reassigned to Judge Leonard Davis for all further proceedings. Signed by Judge William M. Steger on 01/20/05. cc:attys 1-20-05(mll,) (Entered: 01/20/2005)
01/24/2005	6	CORPORATE DISCLOSURE STATEMENT filed by Mosaid Technologies Inc (Jones, Michael) (Entered: 01/24/2005)
02/25/2005	7	NOTICE by Mosaid Technologies Inc *of Dismissal Without Prejudice Pursuant to Federal Rule of Civil Procedure 41(a)(i)* (Jones, Michael) (Entered: 02/25/2005)
05/26/2006	8	NOTICE by Hynix Semiconductor Inc, Hynix Semiconductor America Inc, Hynix Semiconductor Manufacturing America, Inc *of Withdrawal of Cindy M. Allen* (Jones, Michael) (Entered: 05/26/2006)

PACER Service Center			
Transaction Receipt			
07/14/2008 17:41:10			
PACER Login:	fb0051	**Client Code:**	23129-tpl
Description:	Docket Report	**Search Criteria:**	6:05-cv-00013-LED
Billable Pages:	1	**Cost:**	0.08

EXHIBIT E

TO THE DECLARATION OF JEFFREY M. FISHER ISO DEFENDANTS' REPLY ISO MOTION TO DISMISS OR, IN THE ALTERNATIVE, TO TRANSFER VENUE

CLOSED, PATENT

U.S. District Court [LIVE]
Eastern District of TEXAS (Tyler)
CIVIL DOCKET FOR CASE #: 6:05-cv-00120-LED

Mosaid Technologies Inc v. Infineon Technologies North America Corp et al
Assigned to: Judge Leonard Davis
Cause: 35:271 Patent Infringement

Date Filed: 04/06/2005
Date Terminated: 06/20/2006
Jury Demand: Both
Nature of Suit: 830 Patent
Jurisdiction: Federal Question

Mediator

Michael Philip Patterson

Plaintiff

Mosaid Technologies Inc

represented by **Henry Charles Bunsow**
Howrey Simon Arnold & White
525 Market Street
Suite 3600
San Francisco, CA 94105-2708
415/848-4946
Fax: 14158484999
Email: bunsowh@howrey.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Kenneth Robert Adamo
Jones Day - Dallas
2727 North Harwood Street
Dallas, Tx 75201
214-969-4856
Email: kradamo@jonesday.com
TERMINATED: 03/17/2006
LEAD ATTORNEY

Michael Edwin Jones
Potter Minton PC
110 N College
Suite 500
PO Box 359
Tyler, TX 75710-0359
903/597/8311
Fax: 9035930846
Email: mikejones@potterminton.com
TERMINATED: 04/06/2006

LEAD ATTORNEY

Allen Franklin Gardner
Potter Minton PC
110 N College
Suite 500
PO Box 359
Tyler, TX 75710-0359
903/597-8311
Email: allengardner@potterminton.com

TERMINATED: 05/22/2006

Deborah J Race
Ireland Carroll & Kelley
6101 S Broadway
Suite 500
Tyler, TX 75703
903/561-1600
Email: drace@icklaw.com
ATTORNEY TO BE NOTICED

Franklin Jones, Jr
Jones & Jones - Marshall
201 W Houston St
PO Drawer 1249
Marshall, TX 75670
903/938-4395
Fax: 9039383360
Email: maizieh@millerfirm.com
ATTORNEY TO BE NOTICED

James L Wamsley, III
Jones Day - Ohio
901 Lakeside Ave
Cleveland, Oh 44114
216/586-7251
Fax: 216-579-0212
Email: jlwamsleyiii@jonesday.com
TERMINATED: 03/17/2006

Jenny L Sheaffer
Jones Day - Ohio
901 Lakeside Ave
Cleveland, Oh 44114
216/586-7342
Fax: 216/579-0212
Email: jlsheaffer@jonesday.com
TERMINATED: 03/17/2006

John Frederick Bufe
Potter Minton
P. O. Box 359
Tyler, TX 75710
903/597/8311
Fax: 9035930846
Email: johnbufe@potterminton.com
TERMINATED: 04/06/2006

Keith Bryan Davis
Jones Day - Dallas
2727 North Harwood Street
Dallas, Tx 75201
214/969-4528
Email: kbdavis@jonesday.com
TERMINATED: 03/17/2006

Kfir B Levy
Fish & Richardson PC - Washington DC
1425 K St, NW
Suite 1100
Washington, DC 20005
202/626-6358
Fax: 202/783-2331
Email: kylevy@fr.com
ATTORNEY TO BE NOTICED

Korula T Cherian
Howrey LLP - San Francisco
525 Market Street
Suite 3600
San Francisco, Ca 94105
415/848-4946
Fax: 14158484999
Email: cheriank@howrey.com
ATTORNEY TO BE NOTICED

Otis W Carroll, Jr
Ireland Carroll & Kelley
6101 S Broadway
Suite 500
Tyler, TX 75703
903/561-1600
Fax: 9035811071
Email: Fedserv@icklaw.com
ATTORNEY TO BE NOTICED

Patricia L Peden
Law Offices of Patricia L. Peden

5901 Christie Ave
Ste 201
Emeryville, CA 94608
510-268-8033
Email: ppeden@pedenlawfirm.com
ATTORNEY TO BE NOTICED

Robert Scott Wales
Howrey Simon Arnold & White - San Francisco
525 Market Street
Suite 3600
San Francisco, CA 94105-2708
415/848-4987
Fax: 14158484900
Email: waless@howrey.com
ATTORNEY TO BE NOTICED

Scott Wesley Burt
Jones Day
77 West Wacker Drive
Ste 3500
Chicago, IL 60601
312-269-4266
Fax: 13127828585
Email: swburt@jonesday.com
TERMINATED: 03/17/2006

Sean P Hodge
bad address
525 Market Street
Suite 3600
San Francisco, CA 94105-2708
415/848-4900
Fax: 14158484999
ATTORNEY TO BE NOTICED

Sidney Calvin Capshaw, III
Capshaw DeRieux, LLP
1127 Judson Road
Ste 220
Longview, TX 75601-5157
903/233-4826
Fax: 903-236-8787
Email: ccapshaw@capshawlaw.com
ATTORNEY TO BE NOTICED

Thomas John Ward, Jr
WARD & SMITH LAW FIRM
P O Box 1231

Longview, TX 75606-1231
903/757-6400
Fax: 903/7572323
Email: jw@jwfirm.com
ATTORNEY TO BE NOTICED

V.

Defendant

Infineon Technologies North America Corp represented by

Christian Chadd Taylor
Kirkland & Ellis LLP - California
555 California St
Floor 24
San Francisco, CA 94104
415/439-1823
Fax: 415/439-1500
Email: ctaylor@kirkland.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jennifer Parker Ainsworth
Wilson Sheehy Knowles Robertson & Cornelius PC
909 ESE Loop 323
Suite 400
P.O. Box 7339
Tyler, TX 75711-7339
903-509-5000
Fax: 903-509-5092
Email: jainsworth@wilsonlawfirm.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Paul A Bondor
Kirkland & Ellis - NYC
153 E 53rd St
Citicorp Center
New York, NY 10022-4675
212/446-4823
Fax: 12124464900
Email: pbondor@kirkland.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Alice Catherine Garber
Kirkland & Ellis LLP - California
555 California St
Floor 24
San Francisco, CA 94104
415/439-1452

Fax: 415/439-1352
Email: agarber@kirkland.com
ATTORNEY TO BE NOTICED

David Rokach
Kirkland & Ellis LLP - Chicago
200 East Randolph Drive
54th Floor
Chicago, Il 60601
312/861-3169
Fax: 312/861-2200
Email: drokach@kirkland.com
ATTORNEY TO BE NOTICED

Eric M. Albritton
Attorney at Law
PO Box 2649
Longview, TX 75606
903/757-8449
Fax: 19037587397
Email: ema@emafirm.com
ATTORNEY TO BE NOTICED

Gregory S Arovas
Kirkland & Ellis - NYC
153 E 53rd St
Citicorp Center
New York, NY 10022-4675
212/446-4766
Fax: 212/446-4900
Email: garovas@kirkland.com
ATTORNEY TO BE NOTICED

Defendant

Infineon Technologies AG represented by **Christian Chadd Taylor**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jennifer Parker Ainsworth
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Paul A Bondor
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Alice Catherine Garber

(See above for address)
ATTORNEY TO BE NOTICED

David Rokach
(See above for address)
ATTORNEY TO BE NOTICED

Eric M. Albritton
(See above for address)
ATTORNEY TO BE NOTICED

Gregory S Arovas
(See above for address)
ATTORNEY TO BE NOTICED

James A Shimota
Kirkland & Ellis LLP - Chicago
200 East Randolph Drive
54th Floor
Chicago, Il 60601
312/861-2326
Fax: 13128612200
Email: jshimota@kirkland.com
ATTORNEY TO BE NOTICED

Thomas D Pease
Kirkland & Ellis - NYC
153 E 53rd St
Citicorp Center
New York, NY 10022-4675
212/446-4679
Fax: 212/446-4900
Email: tpease@kirkland.com
ATTORNEY TO BE NOTICED

Defendant

Infineon Technologies Holding North America Inc represented by **Christian Chadd Taylor**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jennifer Parker Ainsworth
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Paul A Bondor
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Alice Catherine Garber
(See above for address)
ATTORNEY TO BE NOTICED

David Rokach
(See above for address)
ATTORNEY TO BE NOTICED

Eric M. Albritton
(See above for address)
ATTORNEY TO BE NOTICED

Gregory S Arovas
(See above for address)
ATTORNEY TO BE NOTICED

Defendant

Infineon Technologies Richmond, LP represented by **Christian Chadd Taylor**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jennifer Parker Ainsworth
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Paul A Bondor
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Alice Catherine Garber
(See above for address)
ATTORNEY TO BE NOTICED

David Rokach
(See above for address)
ATTORNEY TO BE NOTICED

Eric M. Albritton
(See above for address)
ATTORNEY TO BE NOTICED

Gregory S Arovas
(See above for address)
ATTORNEY TO BE NOTICED

Counter Claimant

Infineon Technologies North America Corp	represented by	**Jennifer Parker Ainsworth** (See above for address) *LEAD ATTORNEY* *ATTORNEY TO BE NOTICED*

Counter Claimant

Infineon Technologies AG

Counter Claimant

Infineon Technologies Holding North America Inc

Counter Claimant

Infineon Technologies Richmond, LP

V.

Counter Defendant

Mosaid Technologies Inc	represented by	**James L Wamsley, III** (See above for address) *TERMINATED: 03/17/2006* **Jenny L Sheaffer** (See above for address) *TERMINATED: 03/17/2006* **Scott Wesley Burt** (See above for address) *TERMINATED: 03/17/2006*

Counter Claimant

Infineon Technologies North America Corp

Counter Claimant

Infineon Technologies AG

Counter Claimant

Infineon Technologies Holding North America Inc

Counter Claimant

Infineon Technologies Richmond, LP

V.

Counter Defendant

Mosaid Technologies Inc

Date Filed	#	Docket Text
04/06/2005	1	COMPLAINT against Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc, Infineon Technologies Richmond, LP , filed by Mosaid Technologies Inc. (Attachments: # 1 Civil Cover Sheet)(mll,) (Entered: 04/06/2005)
04/06/2005	2	DEMAND for Trial by Jury by Mosaid Technologies Inc. (mll,) (Entered: 04/06/2005)
04/06/2005		Filing fee: $ 250, receipt number 6-607 (mll,) (Entered: 04/06/2005)
04/06/2005	3	NOTICE OF CASE ASSIGNMENT cc:pltf 4-6-05 (mll,) (Entered: 04/06/2005)
04/06/2005	4	Form mailed to Commissioner of Patents and Trademarks. (mll,) (Entered: 04/06/2005)
04/07/2005		Summons Issued as to Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc, Infineon Technologies Richmond, LP. (mll,) (Entered: 04/08/2005)
04/11/2005	5	CORPORATE DISCLOSURE STATEMENT filed by Mosaid Technologies Inc (Jones, Michael) (Entered: 04/11/2005)
04/11/2005	6	Return of Service Executed as to Infineon Technologies Holding North America Inc on 4/8/2005 by personal service; answer due: 4/28/2005. (mll,) (Entered: 04/12/2005)
04/14/2005	7	SUMMONS Returned Executed by Mosaid Technologies Inc. Infineon Technologies Richmond, LP served on 4/8/2005, answer due 4/28/2005. (fnt,) (Entered: 04/15/2005)
04/19/2005	8	ORDER OF RECUSAL. Judge William M. Steger recused. Case reassigned to Judge Leonard Davis for all further proceedings. Signed by Judge William M. Steger on 04/19/05. cc:attys 4-20-05(mll,) (Entered: 04/20/2005)
04/28/2005	9	Consent MOTION for Extension of Time to File Answer *to Plaintiff's Original Complaint* by Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc, Infineon Technologies Richmond, LP. (Ainsworth, Jennifer) Additional attachment(s) added on 4/28/2005 (fnt,). (Entered: 04/28/2005)
04/29/2005	10	ORDER granting 9 Motion for Extension of Time to Answer re 1 Complaint. The response to Plaintiff's original Complaint is extended up to and including June 15, 2005. Signed by Judge Leonard Davis on 4/29/05. (fnt,) (Entered: 04/29/2005)
04/29/2005		Reset Deadlines: Infineon Technologies North America Corp answer due 6/15/2005; Infineon Technologies AG answer due 6/15/2005; Infineon Technologies Holding North America Inc answer due 6/15/2005; Infineon Technologies Richmond, LP answer due 6/15/2005. (fnt,) (Entered: 04/29/2005)

05/16/2005	11	NOTICE of Attorney Appearance by Scott Wesley Burt on behalf of Mosaid Technologies Inc (Burt, Scott) (Entered: 05/16/2005)
05/16/2005	12	NOTICE by Mosaid Technologies Inc *Notice of Appearance of Counsel For Mosaid Technologies, Inc.* (Lanier, Tharan) (Entered: 05/16/2005)
06/15/2005	13	ANSWER to Complaint with Jury Demand, COUNTERCLAIM against Mosaid Technologies Inc by Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc, Infineon Technologies Richmond, LP.(Ainsworth, Jennifer) (Entered: 06/15/2005)
06/21/2005	14	APPLICATION to Appear Pro Hac Vice by Attorney Christian Chadd Taylor for Infineon Technologies Holding North America Inc; Infineon Technologies Richmond, LP; Infineon Technologies North America Corp and Infineon Technologies AG. (mll,) (Entered: 06/22/2005)
06/21/2005	15	APPLICATION to Appear Pro Hac Vice by Attorney Paul A Bondor for Infineon Technologies Holding North America Inc; Infineon Technologies Richmond, LP; Infineon Technologies North America Corp and Infineon Technologies AG. (mll,) (Entered: 06/22/2005)
06/22/2005		Pro Hac Vice Filing fee paid by Christian Taylor and Paul Bondor; Fee: $50.00, receipt number: 6-1412 (mll,) (Entered: 06/22/2005)
06/23/2005	16	ORDER setting Scheduling Conference for 7/20/2005 03:30 PM before Judge Leonard Davis. Signed by Judge Leonard Davis on 06/23/05. cc:attys 6-23-05 (mll,) (Entered: 06/23/2005)
06/27/2005	17	APPLICATION to Appear Pro Hac Vice by Attorney David Rokach for Infineon Technologies Holding North America Inc; Infineon Technologies Richmond, LP; Infineon Technologies North America Corp and Infineon Technologies AG. (fnt,) (Entered: 06/28/2005)
06/27/2005		Pro Hac Vice Filing fee paid by David Rokach; Fee: $25, receipt number: 6-1-001482 (fnt,) (Entered: 06/28/2005)
06/29/2005	18	RESPONSE to 13 Answer to Complaint,, Counterclaim, by Mosaid Technologies Inc. (Jones, Michael) (Entered: 06/29/2005)
06/29/2005	19	APPLICATION to Appear Pro Hac Vice by Attorney Gregory S Arovas for Infineon Technologies Holding North America Inc; Infineon Technologies Richmond, LP; Infineon Technologies North America Corp and Infineon Technologies AG. (fnt,) (Entered: 06/29/2005)
06/29/2005		Pro Hac Vice Filing fee paid by Gregory S Arovas; Fee: $25, receipt number: 6-1-001511 (fnt,) (Entered: 06/29/2005)
07/14/2005	20	APPLICATION to Appear Pro Hac Vice by Attorney Alice C Garber for Infineon Technologies Holding North America Inc; Infineon Technologies Richmond, LP; Infineon Technologies North America Corp and Infineon Technologies AG. Approved 7/15/05 (fnt,) (Entered: 07/15/2005)
07/14/2005		Pro Hac Vice Filing fee paid by Alice C. Garber; Fee: $25, receipt number: 6-

		1-1671 (fnt,) (Entered: 07/15/2005)
07/15/2005	21	MOTION to Change Venue by Infineon Technologies North America Corp, Infineon Technologies North America Corp. (Attachments: # 1 Exhibit # 2 Exhibit # 3 Exhibit # 4 Exhibit # 5 Exhibit # 6 Exhibit # 7 Exhibit # 8 Exhibit # 9 Exhibit # 10 Exhibit # 11 Exhibit # 12 Exhibit # 13 Exhibit # 14 Exhibit # 15 Exhibit # 16 Exhibit # 17 Exhibit # 18 Exhibit # 19 Exhibit # 20 Exhibit # 21 Exhibit # 22 Exhibit # 23 Exhibit # 24 Exhibit # 25 Text of Proposed Order)(Ainsworth, Jennifer) (Entered: 07/15/2005)
07/20/2005	22	Minute Entry for proceedings held before Judge Leonard Davis : Scheduling Conference held on 7/20/2005. (Court Reporter Shea Sloan.) (rlf,) (Entered: 07/21/2005)
07/26/2005	23	ORDER REFERRING CASE to Mediator. Michael Philip Patterson added as Mediator. Signed by Judge Leonard Davis on 7/26/05. (fnt,) (Entered: 07/26/2005)
07/28/2005	24	MOTION for Extension of Time to File Response/Reply as to 21 MOTION to Change Venue *(Unopposed)* by Mosaid Technologies Inc. (Attachments: # 1 Text of Proposed Order)(Jones, Michael) (Entered: 07/28/2005)
07/29/2005	25	DISCOVERY ORDER . Signed by Judge Leonard Davis on 7/28/05. (fnt,) (Entered: 07/29/2005)
07/29/2005	26	ORDER granting 24 Motion for Extension of Time to File Response/Reply Responses due by 8/4/2005. Signed by Judge Judith K. Guthrie on 7/29/05. (fnt,) (Entered: 07/29/2005)
08/01/2005	27	SCHEDULING ORDER: Amended Pleadings due by 7/14/2006. Discovery due by 7/3/2006. Expert Witness List due by 7/6/2006; Parties with burden of proof designate expert witnesses (non-construction issues) due 5/22/06. Identify trial witnesses by 7/14/2006 Joinder of Parties due by 7/29/2005. Jury instructions due by 8/11/2006 Jury Selection set for 10/2/2006 09:00AM before Judge Leonard Davis. Mediation Completion due by 2/24/2006. Motions due by 7/28/2006. Proposed Findings of Fact due by 8/11/2006 Proposed Pretrial Order due by 8/11/2006. Signed by Judge Leonard Davis on 7/28/05. (fnt,) Modified on 8/1/2005 (fnt,). (Entered: 08/01/2005)
08/01/2005		Set Deadlines/Hearings: continuation of # 27 Exhibit List due by 10/10/2006. Jury Trial set for 10/10/2006 09:00 AM before Judge Leonard Davis.Markman Hearing set for 4/6/2006 09:00 AM before Judge Leonard Davis. (fnt,) (Entered: 08/01/2005)
08/01/2005	28	NOTICE of Disclosure by Mosaid Technologies Inc *of Asserted Claims and Preliminary Infringement Contentions* (Jones, Michael) (Entered: 08/01/2005)
08/04/2005	29	RESPONSE in Opposition re 21 MOTION to Change Venue filed by Mosaid Technologies Inc. (Attachments: # 1 Exhibit 1# 2 Exhibit 2# 3 Exhibit 3# 4 Exhibit 4# 5 Exhibit 5# 6 Exhibit 6# 7 Exhibit 7# 8 Exhibit 8# 9 Exhibit 9# 10 Exhibit 10# 11 Exhibit 11# 12 Exhibit 12# 13 Exhibit 13# 14 Exhibit 14# 15 Exhibit 15# 16 Exhibit 16# 17 Exhibit 17# 18 Exhibit 18# 19 Exhibit 19# 20 Exhibit 20# 21 Exhibit 21# 22 Exhibit 22# 23 Exhibit 23)(Jones, Michael)

		(Entered: 08/04/2005)
08/09/2005	30	NOTICE of Attorney Appearance by Allen Franklin Gardner on behalf of Mosaid Technologies Inc (Gardner, Allen) (Entered: 08/09/2005)
08/11/2005	31	MOTION for Leave to File *Amended Disclosure of Asserted Claims and Preliminary Infringement Contentions and to Modify the Docket Control Order (Unopposed)* by Mosaid Technologies Inc. (Attachments: # 1 Exhibit A# 2 Text of Proposed Order)(Jones, Michael) (Entered: 08/11/2005)
08/11/2005	32	MOTION for Leave to File Excess Pages *on Reply in Support of Motion for Change of Venue UNOPPOSED* by Infineon Technologies North America Corp. (Attachments: # 1 Text of Proposed Order)(Ainsworth, Jennifer) (Entered: 08/11/2005)
08/11/2005	33	TRANSCRIPT of Scheduling Conference Proceedings held on 7/20/05, 3:32 PM, before Judge Leonard Davis. Court Reporter: Shea Sloan. (fnt,) (Entered: 08/11/2005)
08/11/2005	34	Sealed Document. Infineon Technologies North America Corporation's Reply (Attachments: # 1 Exhibit A thru Exhibit AB# 2 Exhibit AC# 3 Exhibit AD thru Exhibit AE# 4 Exhibit AF thru Exhibit AK# 5 Exhibit AL thru Exhibit AQ# 6 Exhibit AR thru Exhibit AV)(fnt,) (Entered: 08/12/2005)
08/12/2005	35	APPLICATION to Appear Pro Hac Vice by Attorney Thomas D Pease for Infineon Technologies AG. Approved 8/12/05, FT(fnt,) (Entered: 08/12/2005)
08/12/2005		Pro Hac Vice Filing fee paid by Thomas D. Pease; Fee: $25, receipt number: 6-1-2001 (fnt,) (Entered: 08/12/2005)
08/12/2005	36	ORDER granting 31 Motion for Leave to Amend Disclosure of Asserted claims and Preliminary Infringement contentions and to Modify the Docket Control Order. Signed by Judge Leonard Davis on 8/12/05. (fnt,) (Entered: 08/12/2005)
08/12/2005	37	ORDER granting 32 Motion for Leave to File Excess Pages . Signed by Judge Leonard Davis on 8/12/05. (fnt,) (Entered: 08/12/2005)
08/15/2005	38	MOTION to Strike 34 Sealed Document, *Infineon's Reply to its Motion for a Change of Venue to the Northern District of California Pursuant to 28 USC Sec. 1404(A)* by Mosaid Technologies Inc. (Attachments: # 1 Text of Proposed Order)(Jones, Michael) (Entered: 08/15/2005)
08/16/2005	39	NOTICE of Disclosure by Mosaid Technologies Inc *and Amended Preliminary Infringement Contentions* (Jones, Michael) (Entered: 08/16/2005)
08/16/2005	41	APPLICATION to Appear Pro Hac Vice by Attorney James L Wamsley for Mosaid Technologies Inc and Mosaid Technologies Inc.(Application Approved 8/16/2005) (rvw,) (Entered: 08/17/2005)
08/16/2005	42	APPLICATION to Appear Pro Hac Vice by Attorney Jenny L Sheaffer for Mosaid Technologies Inc and Mosaid Technologies Inc. (Application Approved 8/16/2005) (rvw,) (Entered: 08/17/2005)

08/16/2005		Pro Hac Vice Filing fee paid by Wamsley and Sheaffer; Fee: $50.00, receipt number: 6-1-2033 (rvw,) (Entered: 08/17/2005)
08/17/2005	40	"DOCUMENT FILED IN ERROR, SEE DOC 43 FOR CORRECTED DOCUMENT" MOTION to Amend/Correct *Agreed Motion to Enter Addendum Supplementing Protective Order* by Mosaid Technologies Inc, Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc, Infineon Technologies Richmond, LP. (Attachments: # 1 Exhibit A - Addendum Supplementing Protective Order)(Jones, Michael) Modified on 8/17/2005 (rvw,). (Entered: 08/17/2005)
08/17/2005	43	"REPLACES DOCUMENT 40 WHICH WAS FILED IN ERROR" MOTION to Amend/Correct *Agreed Motion to Enter Addendum Supplementing Protective Order* by Mosaid Technologies Inc, Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc, Infineon Technologies Richmond, LP. (Attachments: # 1 Exhibit A - Proposed Addendum# 2 Exhibit A to Proposed Addendum# 3 Exhibit B to Proposed Addendum)(Jones, Michael) Modified on 8/17/2005 (rvw,). (Entered: 08/17/2005)
08/19/2005	44	ADDENDUM SUPPLEMENTING PROTECTIVE ORDER. Signed by Judge David Folsom on 8/19/05. (fnt,) (Entered: 08/19/2005)
08/19/2005	45	MOTION for Leave to File Excess Pages *for MOSAID'S Sur-reply to Infineon's Motion for a Change of Venue to the Northern District of California Pursuant to 28 USC Sec. 1404(a) [UNOPPOSED]* by Mosaid Technologies Inc. (Attachments: # 1 Text of Proposed Order)(Jones, Michael) (Entered: 08/19/2005)
08/19/2005	46	REPLY to Response to Motion re 21 MOTION to Change Venue filed by Mosaid Technologies Inc. (Attachments: # 1 Exhibit 1 - Taylor Declaration# 2 Exhibit 2 - Cirit Declaration)(Jones, Michael) (Entered: 08/19/2005)
08/19/2005	47	RESPONSE in Opposition re 38 MOTION to Strike 34 Sealed Document, *Infineon's Reply to its Motion for a Change of Venue to the Northern District of California Pursuant to 28 USC Sec. 1404(A)* filed by Infineon Technologies North America Corp. (Ainsworth, Jennifer) (Entered: 08/19/2005)
08/19/2005	48	NOTICE of Disclosure by Mosaid Technologies Inc *(Initial Disclosures)* (Jones, Michael) (Entered: 08/19/2005)
08/25/2005	49	ORDER granting 45 Motion for Leave to File Excess Pages. ORDERED that the page limit for Plaintiff's Sur-Reply to Infineon's Motion for a Change of Venue to the Norhtern District of California Pursuant to 28 USC 1404(A) should be expanded by 14 pages to allow 19 pages . Signed by Judge Leonard Davis on 8/25/05. (fnt,) Modified on 8/25/2005 (fnt,). (Entered: 08/25/2005)
10/04/2005	50	NOTICE of Disclosure by Mosaid Technologies Inc *of Proposed Terms and Claim Elements for Construction* (Jones, Michael) (Entered: 10/04/2005)
10/06/2005	51	NOTICE by Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc,

		Infineon Technologies Richmond, LP *of Disclosure of Proposed Terms and Claim Elements for Construction Pursuant to Patent Rule 4-1* (Garber, Alice) (Entered: 10/06/2005)
10/17/2005	52	ORDER denying 21 Motion to Change Venue, denying 38 Motion to Strike . Signed by Judge Leonard Davis on 10/15/05. (fnt,) (Entered: 10/17/2005)
10/19/2005	53	NOTICE of Attorney Appearance by John Frederick Bufe on behalf of Mosaid Technologies Inc (Bufe, John) (Entered: 10/19/2005)
10/31/2005	54	NOTICE of Attorney Appearance by Eric Miller Albritton on behalf of Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc, Infineon Technologies Richmond, LP (Albritton, Eric) (Entered: 10/31/2005)
12/06/2005	55	NOTICE of Disclosure by Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc, Infineon Technologies Richmond, LP *of Exchange of Preliminary Claim Constructions and Extrinsic Evidence and Privilege Log* (Garber, Alice) (Entered: 12/06/2005)
12/06/2005	56	NOTICE of Disclosure by Mosaid Technologies Inc *of Exchange of Preliminary Claim Constructions and Extrinsic Evidence and Privilege Log* (Davis, Keith) (Entered: 12/06/2005)
01/06/2006	57	NOTICE by Mosaid Technologies Inc *P.R. 4-3 Joint Claim Construction and Prehearing Statement* (Attachments: # 1 Exhibit A# 2 Exhibit B# 3 Exhibit C) (Davis, Keith) (Entered: 01/06/2006)
01/17/2006	58	NOTICE by Mosaid Technologies Inc *MOSAID's Disclosure of Techical Advisors* (Attachments: # 1 Exhibit A# 2 Exhibit B# 3 Exhibit C)(Davis, Keith) (Entered: 01/17/2006)
01/17/2006	59	NOTICE of Disclosure by Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc, Infineon Technologies Richmond, LP *of Technical Advisors* (Attachments: # 1 Exhibit A)(Garber, Alice) (Entered: 01/17/2006)
01/18/2006	64	APPLICATION to Appear Pro Hac Vice by Attorney James A Shimota for Infineon Technologies AG. Approved 1/23/05(fnt,) (Entered: 01/23/2006)
01/18/2006		Pro Hac Vice Filing fee paid by Shimota; Fee: $25, receipt number: 611083 (fnt,) (Entered: 01/23/2006)
01/19/2006	60	ORDER: The Court is considering Lou Brucculeri of Wong, Cabello, Lutsch, Rutherford & Brucculeri in Houston, TX for the position of technical advisor in this case. If the parties have any objections to Mr. Brucculeri's appointment, they are to file their objections for in camera review no later than January 31, 2006.Objections due by 1/31/2006. Signed by Judge Leonard Davis on 1/19/06. (fnt,) (Entered: 01/19/2006)
01/19/2006	61	Joint MOTION to Amend/Correct 57 Notice (Other) *Exhibits to Joint Claim Construction and Prehearing Statement* by Mosaid Technologies Inc. (Attachments: # 1 Exhibit B# 2 Exhibit C# 3 Text of Proposed Order)(Davis,

		Keith) (Entered: 01/19/2006)
01/20/2006	62	ORDER granting 61 Motion to Amend/Correct Exhibits to Joint Claim Construction and Prehearing Statement. It is therefore ORDERED that Exhibits B and C attached to the Motion are entered as Exhibits B and C to the Parties' Joint Claim Construction and Pre-Hearing Statement (Docket No. 57, filed 1/6/06), thereby amending and superseding Exhibits B and C as originally filed. Signed by Judge Leonard Davis on 1/20/06. (fnt,) (Entered: 01/20/2006)
01/20/2006	63	EXHIBIT B and C to the Parties' Joint Claim Construction and Pre-Hearing Statement (Attachments: # 1 Exhibit C)(fnt,) (Entered: 01/20/2006)
01/31/2006	65	Consent MOTION to Amend/Correct 1 Complaint, by Mosaid Technologies Inc. (Attachments: # 1 Affidavit Declaration of Keith B. Davis# 2 Exhibit A# 3 Exhibit B# 4 Exhibit C# 5 Exhibit D# 6 Exhibit E# 7 Exhibit F# 8 Exhibit G# 9 Exhibit H# 10 Exhibit I# 11 Exhibit J)(Davis, Keith) (Entered: 01/31/2006)
02/01/2006	66	*****FILED IN ERROR, PLEASE IGNORE***** Additional Attachments to Main Document: 65 Consent MOTION to Amend/Correct 1 Complaint,.. (Davis, Keith) Modified on 2/3/2006 (fnt,). (Entered: 02/01/2006)
02/02/2006	67	ORDER granting 65 Motion to Amend/Correct . Signed by Judge Leonard Davis on 2/1/06. (fnt,) (Entered: 02/02/2006)
02/02/2006	68	ORDER appointing Lou Brucculeri of Wong, Cabello, Lutsch, Rutherford & Brucculeri in Houston, Texas to the position of technical advisor in this case. Signed by Judge Leonard Davis on 2/2/06. (fnt,) (Entered: 02/02/2006)
02/07/2006	69	MOTION to Amend/Correct *Preliminary Invalidity Contentions* by Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc, Infineon Technologies Richmond, LP. (Garber, Alice) (Entered: 02/07/2006)
02/07/2006	70	Consent MOTION to Amend/Correct *Preliminary Infringement Contentions* by Mosaid Technologies Inc. (Attachments: # 1 Text of Proposed Order) (Davis, Keith) (Entered: 02/07/2006)
02/08/2006	71	ORDER granting 69 Motion to Amend/Correct its Preliminary Invalidity Contention. Signed by Judge Leonard Davis on 2/8/06. (fnt,) (Entered: 02/08/2006)
02/08/2006	72	ORDER granting 70 Motion to Amend/Correct Unopposed P.R. 3-7 Motion for leave to supplement its Preliminary Infringement Contentions. Signed by Judge Leonard Davis on 2/8/06. (fnt,) (Entered: 02/08/2006)
02/15/2006	73	NOTICE by Mosaid Technologies Inc *Notice of Submission of Technology Tutorial* (Jones, Michael) (Entered: 02/15/2006)
02/16/2006	74	REPORT of Mediation by Michael Philip Patterson. Mediation result: recessed(Patterson, Michael) (Entered: 02/16/2006)

02/22/2006	75	TRIAL BRIEF *Opening Claim Construction Brief* by Mosaid Technologies Inc. (Attachments: # 1 Affidavit Declaration of Keith B. Davis# 2 Exhibit A# 3 Exhibit B# 4 Exhibit C# 5 Exhibit D# 6 Exhibit E# 7 Exhibit F# 8 Exhibit G# 9 Exhibit H# 10 Exhibit I# 11 Exhibit J# 12 Exhibit K# 13 Exhibit L# 14 Exhibit M# 15 Affidavit Declaration of Richard Greene) Exhibit M continue# Curriculum Vitae(Davis, Keith) (Entered: 02/22/2006)
03/08/2006	76	NOTICE of Attorney Appearance by Robert Scott Wales on behalf of Mosaid Technologies Inc (Wales, Robert) (Entered: 03/08/2006)
03/08/2006	77	Joint MOTION to Amend/Correct 27 Scheduling Order,, by Mosaid Technologies Inc, Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc, Infineon Technologies Richmond, LP. (Attachments: # 1 Text of Proposed Order)(Albritton, Eric) (Entered: 03/08/2006)
03/09/2006	78	NOTICE of Attorney Appearance by Henry Charles Bunsow on behalf of Mosaid Technologies Inc (Bunsow, Henry) (Entered: 03/09/2006)
03/09/2006	79	ORDER denying 77 Motion to Amend Docket Control Order. Signed by Judge Leonard Davis on 3/9/2006. (rvw,) (Entered: 03/09/2006)
03/09/2006	80	NOTICE of Attorney Appearance by Korula T Cherian on behalf of Mosaid Technologies Inc (Cherian, Korula) (Entered: 03/09/2006)
03/10/2006	81	"PLEASE IGNORE, FILED IN ERROR, SEE DOC# 82 FOR CORRECTION" Consent MOTION for Reconsideration re 79 Order on Motion to Amend/Correct by Mosaid Technologies Inc. (Attachments: # 1) (Carroll, Otis) Modified on 3/10/2006 (rvw,). (Entered: 03/10/2006)
03/10/2006	82	"CORRECTED DOCUMENT, REPLACES DOC 81 " Consent MOTION for Reconsideration re 79 Order on Motion to Amend/Correct by Mosaid Technologies Inc. (Attachments: # 1 Text of Proposed Order)(Carroll, Otis) Modified on 3/10/2006 (rvw,). (Entered: 03/10/2006)
03/13/2006	83	NOTICE of Attorney Appearance by Patricia L Peden on behalf of Mosaid Technologies Inc (Peden, Patricia) (Entered: 03/13/2006)
03/13/2006	84	MOTION for Extension of Time to File *Unopposed Motion to Extend Deadline to File Markman Brief Pending Resolution of Motion for Reconsideration* by Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc, Infineon Technologies Richmond, LP. (Attachments: # 1 Text of Proposed Order)(Albritton, Eric) (Entered: 03/13/2006)
03/13/2006	85	NOTICE of Attorney Appearance by Thomas John Ward, Jr on behalf of Mosaid Technologies Inc (Ward, Thomas) (Entered: 03/13/2006)
03/14/2006	86	ORDER granting 84 Motion for Extension of Time to File Markman Brief Pending Resolution of Motion for Reconsideration. Signed by Judge Leonard Davis on 3/13/06. (fnt,) (Entered: 03/14/2006)
03/14/2006	87	ORDER granting 82 Motion for Reconsideration . Signed by Judge Leonard Davis on 3/13/06. (fnt,) (Entered: 03/14/2006)

03/14/2006		Set Deadlines/Hearings: Agreed Motions to amend the Docket Control Order due by 3/15/2006. Markman Hearing rescheduled for 5/31/2006 09:00 AM before Judge Leonard Davis. (fnt,) (Entered: 03/14/2006)
03/15/2006	88	Joint MOTION to Amend/Correct 27 Scheduling Order,, by Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc, Infineon Technologies Richmond, LP. (Garber, Alice) (Entered: 03/15/2006)
03/16/2006	89	Consent MOTION to Withdraw as Attorney *JONES DAY* by Mosaid Technologies Inc, Mosaid Technologies Inc. (Attachments: # 1 Text of Proposed Order ORDER WITHDRAWING JONES DAY AS COUNSEL FOR MOSAID)(Burt, Scott) (Entered: 03/16/2006)
03/17/2006	90	ORDER re 88 Joint MOTION to Amend/Correct 27 Scheduling Order,, filed by Infineon Technologies AG,, Infineon Technologies Richmond, LP,, Infineon Technologies North America Corp,, Infineon Technologies Holding North America Inc, Discovery due by 8/4/2006. Expert Witness List due by 6/26/2006.Markman Hearing set for 5/31/2006 09:00 AM before Judge Leonard Davis. Signed by Judge Leonard Davis on 3/17/06. (fnt,) (Entered: 03/20/2006)
03/17/2006	91	ORDER granting 89 Motion to Withdraw as Attorney; Withdrawing Jones Day as counsel for Mosaid. Attorney Keith Bryan Davis; Jenny L Sheaffer; James L Wamsley; Kenneth Robert Adamo and Scott Wesley Burt terminated . Signed by Judge Leonard Davis on 3/17/06. (fnt,) (Entered: 03/20/2006)
03/22/2006	92	NOTICE of Attorney Appearance by Franklin Jones, Jr on behalf of Mosaid Technologies Inc (Jones, Franklin) (Entered: 03/22/2006)
03/23/2006	93	Joint MOTION to Amend/Correct *the Protective Order* by Mosaid Technologies Inc. (Attachments: # 1 Text of Proposed Order)(Wales, Robert) (Entered: 03/23/2006)
03/23/2006	94	APPLICATION to Appear Pro Hac Vice by Attorney Sean P Hodge for Mosaid Technologies Inc. Approved 3/24/06(fnt,) (Entered: 03/24/2006)
03/23/2006		Pro Hac Vice Filing fee paid by Sean P Hodge; Fee: $25, receipt number: 614510 (fnt,) (Entered: 03/24/2006)
03/27/2006	95	ORDER granting 93 Motion to Amend/Correct . Signed by Judge Leonard Davis on 3/27/06. (fnt,) (Entered: 03/27/2006)
03/30/2006	96	NOTICE of Attorney Appearance by Kfir B Levy on behalf of Mosaid Technologies Inc (Levy, Kfir) (Entered: 03/30/2006)
03/30/2006	97	AMENDED COMPLAINT - *First Amended Complaint for Patent Infringement and Jury Demand* against Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc, Infineon Technologies Richmond, LP, filed by Mosaid Technologies Inc.(Wales, Robert) (Entered: 03/30/2006)
04/05/2006	98	MOTION to Withdraw *as Counsel for Plaintiff Mosaid Technologies* by

		Mosaid Technologies Inc. (Attachments: # 1 Text of Proposed Order)(Jones, Michael) (Entered: 04/05/2006)
04/06/2006	99	ORDER granting 98 Motion to Withdraw as counsel for Plaintiff MOSAID Technologies, Inc . Signed by Judge Leonard Davis on 4/6/06. (fnt,) (Entered: 04/06/2006)
04/13/2006	100	MOTION for Extension of Time to File Answer re 97 Amended Complaint, *or Otherwise Respond* by Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc, Infineon Technologies Richmond, LP. (Attachments: # 1 Text of Proposed Order)(Albritton, Eric) (Entered: 04/13/2006)
04/14/2006	101	ORDER granting 100 Motion for Extension of Time to Answer re 97 Amended Complaint. Infineon shall have until and including April 20, 2006, to answer or otherwise respond to the First Amended Complaint . Signed by Judge Leonard Davis on 4/14/06. (fnt,) (Entered: 04/17/2006)
04/14/2006		Answer Due Deadline Updated for Infineon Technologies AG to 4/20/2006. (fnt,) (Entered: 04/17/2006)
04/18/2006	102	MOTION for Leave to File *Second Amended Complaint* by Mosaid Technologies Inc. (Attachments: # 1 Affidavit A)(Wales, Robert) Additional attachment(s) added on 4/19/2006 (fnt,). (Entered: 04/18/2006)
04/18/2006	103	*****FILED IN ERROR, PLEASE IGNORE***** Additional Attachments to Main Document: 102 MOTION for Leave to File *Second Amended Complaint*.. (Wales, Robert) Modified on 4/19/2006 (fnt,). (Entered: 04/18/2006)
04/19/2006	104	ORDER granting 102 Motion for Leave to File Second Amended Complaint. Signed by Judge Leonard Davis on 4/19/06. (fnt,) (Entered: 04/19/2006)
04/20/2006	105	AMENDED COMPLAINT - *Second Amended Complaint for Patent Infringement* against all defendants, filed by Mosaid Technologies Inc. (Attachments: # 1 Exhibit A# 2 Exhibit B# 3 Exhibit C)(Wales, Robert) (Entered: 04/20/2006)
05/02/2006	106	SEALED PATENT DOCUMENT *Infineon's Claim Construction Brief*. (Attachments: # 1 Exhibit A# 2 Exhibit B# 3 Exhibit C# 4 Exhibit D# 5 Exhibit E# 6 Exhibit F# 7 Exhibit G# 8 Exhibit H# 9 Exhibit I# 10 Exhibit J# 11 Exhibit K# 12 Exhibit L# 13 Exhibit M# 14 Exhibit N# 15 Exhibit O# 16 Exhibit P# 17 Exhibit Q# 18 Exhibit R)(Albritton, Eric) (Entered: 05/02/2006)
05/04/2006	107	NOTICE of Attorney Appearance by Deborah J Race on behalf of Mosaid Technologies Inc (Race, Deborah) (Entered: 05/04/2006)
05/04/2006	108	ANSWER to Amended Complaint, COUNTERCLAIM against Mosaid Technologies Inc by Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc, Infineon Technologies Richmond, LP.(Garber, Alice) (Entered: 05/04/2006)
05/11/2006	109	BRIEF filed *Plaintiff MOSAID Technologies, Inc.'s Reply Claim Construction*

		Brief by Mosaid Technologies Inc. (Wales, Robert) (Entered: 05/11/2006)
05/11/2006	110	BRIEF filed *Declaration of Lorrel Birnschein In Support of MOSAID's Reply Claim Construction Brief* by Mosaid Technologies Inc. (Wales, Robert) (Entered: 05/11/2006)
05/11/2006	111	BRIEF filed *Declaration of David Taylor In Support of MOSAID's Response to Infineon's Claim Construction Brief* by Mosaid Technologies Inc. (Wales, Robert) (Entered: 05/11/2006)
05/11/2006	112	BRIEF filed *Supplemental Declaration of Richard Greene In Support of MOSAID's Claim Construction Brief* by Mosaid Technologies Inc. (Wales, Robert) (Entered: 05/11/2006)
05/17/2006	113	NOTICE of Attorney Appearance by Sidney Calvin Capshaw, III on behalf of Mosaid Technologies Inc (Capshaw, Sidney) (Entered: 05/17/2006)
05/17/2006	114	MOTION to Withdraw as Attorney by Mosaid Technologies Inc. (Attachments: # 1 Text of Proposed Order)(Gardner, Allen) (Entered: 05/17/2006)
05/17/2006	115	NOTICE by Mosaid Technologies Inc *(Notice of Compliance with P. R. 4-5 (d))* (Capshaw, Sidney) (Entered: 05/17/2006)
05/22/2006	116	ORDER granting 114 Motion to Withdraw as Attorney. Attorney Allen Franklin Gardner terminated . Signed by Judge Leonard Davis on 5/22/06. (fnt,) (Entered: 05/22/2006)
05/30/2006	117	NOTICE: Per parties' announcement of settlement, parties are directed to file a Motion and Proposed Order for Dismissal within 30 days of the date of this notice, or the Court will Dismiss case with Prejudice on its own motion. cc: Parties. (rlf,) (Entered: 05/30/2006)
05/31/2006	118	ORDER: The Court ORDERS the parties to submit payment to Mr Brucculeri as follows: Plaintiff: $9,887.25; Defendant: $9,887.25, Total: $19,774.50. Signed by Judge Leonard Davis on 5/31/06. (fnt,) (Entered: 05/31/2006)
06/15/2006	119	Joint MOTION to Dismiss *with Prejudice* by Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc, Infineon Technologies Richmond, LP. (Garber, Alice) Additional attachment(s) added on 6/19/2006 (rvw,). (Entered: 06/15/2006)
06/15/2006	120	"FILED IN ERROR, SEE DOCUMENT 119 FOR CORRECT MOTION" Joint MOTION to Dismiss *[Proposed Order on Joint Motion to Dismiss]* by Infineon Technologies North America Corp, Infineon Technologies AG, Infineon Technologies Holding North America Inc, Infineon Technologies Richmond, LP. (Garber, Alice) Modified on 6/19/2006 (rvw,). (Entered: 06/15/2006)
06/20/2006	121	ORDER granting 119 Motion to Dismiss with prejudice. Each party shall bear its own costs and attorney's fees. Signed by Judge Leonard Davis on 6/20/06. (mjc) (Entered: 06/20/2006)

PACER Service Center			
Transaction Receipt			
07/14/2008 17:41:56			
PACER Login:	fb0051	**Client Code:**	23129-tpl
Description:	Docket Report	**Search Criteria:**	6:05-cv-00120-LED
Billable Pages:	11	**Cost:**	0.88

EXHIBIT F

TO THE DECLARATION OF JEFFREY M. FISHER ISO DEFENDANTS' REPLY ISO MOTION TO DISMISS OR, IN THE ALTERNATIVE, TO TRANSFER VENUE

(RCx), AO279, CLOSED, DISCOVERY, PROTORD, RELATED-G

UNITED STATES DISTRICT COURT, CENTRAL DISTRICT OF CALIFORNIA (Western Division - Los Angeles) CIVIL DOCKET FOR CASE #: 2:07-cv-00692-R-RC

Guardian Media Technologies Ltd v. LG Electronics Inc et al
Assigned to: Judge Manuel L. Real
Referred to: Magistrate Judge Rosalyn M. Chapman
Related Case: 2:06-cv-04910-R-RC
Cause: 35:271 Patent Infringement

Date Filed: 01/29/2007
Date Terminated: 01/28/2008
Jury Demand: Both
Nature of Suit: 830 Patent
Jurisdiction: Federal Question

Plaintiff

Guardian Media Technologies Ltd represented by

Brian L Jackson
Howrey
1111 Louisiana, 25th Floor
Houston, TX 77002-5242
713-787-1507
TERMINATED: 09/14/2007
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jan L Handzlik
Howrey LLP
550 South Hope Street Suite 1100
Los Angeles, CA 90071
213-892-1800
Email: handzlikj@howrey.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Michael S Dowler
Howrey
1111 Louisiana, 25th Floor
Houston, TX 77002-5242
713-787-1400
Email: dowlerm@howrey.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Shane Nelson
Howrey
1111 Louisiana, 25th Floor
Houston, TX 77002-5242

713-787-1433
Email: nelsons@howrey.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Stephen L Lundwall
Howrey
1111 Louisiana, 25th Fl
Houston, TX 77002-5242
713-787-1498
Email: lundwalls@howrey.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Thomas L Casagrande
Howrey
1111 Louisiana, 25th Floor
Houston, TX 77002
713-787-1400
Email: casagrandet@howrey.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

V.

Defendant

LG Electronics Inc
TERMINATED: 06/04/2007

Defendant

LG Electronics USA Inc
TERMINATED: 06/04/2007

Defendant

Harsper Co Ltd

Defendant

Harsper USA Inc

Defendant

Santax Groups Corp represented by **Casondra K Ruga**
Weston Benshoof Rochefort Rubalcava MacCuish
333 South Hope Street, 16th Floor
Los Angeles, CA 90071
213-576-1100
Email: cruga@wbcounsel.com
TERMINATED: 10/05/2007
LEAD ATTORNEY

ATTORNEY TO BE NOTICED

Darin Margules
Tyre Kamins Katz & Granof
1880 Century Park E
Ste 300
Los Angeles, CA 90067-1666
310-553-6822
Email: dmargules@tyrekamins.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Herman S Palarz
Tyre Kamins Katz Granof and Menes
1880 Century Park E
Ste 315
Los Angeles, CA 90067-1666
310-553-6822
Fax: 310-552-9024
Email: hpalarz@tyrekamins.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John P Poliak
Pauley Petersen & Erickson
2800 West Higgins Road, Suite 365
Hoffman, IL 60169
847-490-1400
Email: jpoliak@ppelaw.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Maxwell J Petersen
Pauley Peterson & Erickson
2800 West Higgins Road, Suite 365
Hoffman Estates, IL 60169
847-490-1400
Email: mjp@ppelaw.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Defendant

SVA (Group) Co Ltd
TERMINATED: 01/17/2008

represented by **Anthony C Roth**
Morgan Lewis & Bockius
1111 Pennsylvania Ave NW
Washington, DC 20004
202-739-5188
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Lisa Sharrock Glasser
Irell & Manella
840 Newport Center Drive, Suite 400
Newport Beach, CA 92660-6324
949-760-0991
Email: lglasser@irell.com
TERMINATED: 09/28/2007
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Richard Hyungil Lee
Morgan Lewis and Bockius
300 South Grand Avenue, 22nd Floor
Los Angeles, CA 90071
213-612-2500
Email: richard.lee@morganlewis.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Defendant

Sva Group (USA)Inc
TERMINATED: 01/17/2008

represented by **David KW Chang**
David KW Chang Law Offices
660 North Diamond Bar Boulevard
Suite 210
Diamond Bar, CA 91765-1034
909-612-5888
Email: dc@dchanglaw.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Richard Hyungil Lee
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Defendant

Syntax-Brillan Corp

represented by **Casondra K Ruga**
(See above for address)
TERMINATED: 09/12/2007
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Darin Margules
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Herman S Palarz
(See above for address)
LEAD ATTORNEY

ATTORNEY TO BE NOTICED

John P Poliak
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Leah Phillips Falzone
Tyre Kamins Katz Granof and Menes
1880 Century Park East
Suite 300
Los Angeles, CA 90067-1666
310-553-6822
Email: lphillips@tyrekamins.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Maxwell J Petersen
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Counter Claimant

Syntax-Brillan Corp represented by **Casondra K Ruga**
(See above for address)
TERMINATED: 10/05/2007
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Darin Margules
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Herman S Palarz
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

V.

Counter Defendant

Guardian Media Technologies Ltd represented by **Brian L Jackson**
(See above for address)
TERMINATED: 09/14/2007
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jan L Handzlik
(See above for address)

LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Michael S Dowler
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Shane Nelson
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Thomas L Casagrande
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Counter Claimant

Sva Group (USA)Inc
TERMINATED: 01/17/2008

represented by **David KW Chang**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Richard Hyungil Lee
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

V.

Counter Defendant

Guardian Media Technologies Ltd

represented by **Brian L Jackson**
(See above for address)
TERMINATED: 09/14/2007
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jan L Handzlik
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Michael S Dowler
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Shane Nelson
(See above for address)

LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Thomas L Casagrande
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Date Filed	#	Docket Text
01/29/2007	1	COMPLAINT against defendants LG Electronics Inc, LG Electronics USA Inc.(Filing fee $ 350 Paid) Jury Demanded. , filed by plaintiff Guardian Media Technologies Ltd.(rrey,) (Entered: 02/02/2007)
01/29/2007		20 Day Summons Issued re Complaint - (Discovery) 1 as to LG Electronics Inc, LG Electronics USA Inc. (rrey,) (Entered: 02/02/2007)
01/29/2007	2	NOTICE of Interested Parties filed by Plaintiff Guardian Media Technologies Ltd. (rrey,) (Entered: 02/02/2007)
01/29/2007	3	APPLICATION of Brian L Jackson for Leave to Appear Pro Hac Vice. FEE PAID. filed by plaintiff Guardian Media Technologies Ltd. Lodged order. (rrey,) (Entered: 02/02/2007)
01/29/2007	4	APPLICATION of Shane Nelson for Leave to Appear Pro Hac Vice. FEE PAID. filed by plaintiff Guardian Media Technologies Ltd. Lodged order. (rrey,) (Entered: 02/02/2007)
01/29/2007	5	APPLICATION of Michael S Dowler for Leave to Appear Pro Hac Vice. FEE PAID. filed by plaintiff Guardian Media Technologies Ltd. Lodged order. (rrey,) (Entered: 02/02/2007)
01/29/2007	6	APPLICATION of Thomas L Casagrande for Leave to Appear Pro Hac Vice. FEE PAID. filed by plaintiff Guardian Media Technologies Ltd. Lodged order. (rrey,) (Entered: 02/02/2007)
01/29/2007	7	NOTICE TO PARTIES OF ADR PILOT PROGRAM filed.(rrey,) (Entered: 02/02/2007)
01/29/2007		REPORT ON THE FILING OF AN ACTION REGARDING PATENTS (cc: form mailed to Washington, D.C.) (Opening) (rrey,) (Entered: 02/02/2007)
01/29/2007	8	NOTICE of Related Case(s) filed by Plaintiff Guardian Media Technologies Ltd. Related Case(s): CV 06-4910 R (RCx) (rn,) (Entered: 02/05/2007)
02/01/2007	11	INITIAL STANDING ORDER FOR CASES ASSIGNED TO Judge Manuel L. Real, READ THIS ORDER CAREFULLYL. IT CONTROLS THIS CASE AND DIFFERS IN SOME RESPECTS FROM THE LOCAL RULES. (see file)(yc,) (Entered: 02/09/2007)
02/05/2007		PREPARED ORDER RE TRANSFER Pursuant to General Order 224 (Related Case) by Clerk; related to CV 06-4910 R (RCx). (rn,) (Entered: 02/05/2007)

02/06/2007	9	ORDER RE TRANSFER PURSUANT TO GENERAL ORDER 224 -Related Case- filed. Related Case No: CV 06-4910-R(RCx). Case transfered from Judge Audrey B. Collins and Magistrate Judge Charles F. Eick to Judge Manuel L. Real and Magistrate Judge Rosalyn M. Chapman for all further proceedings. The case number will now reflect the initials of the transferee Judge CV 07-692-R(RCx).Signed by Judge Manuel L. Real (stsh,) (Entered: 02/06/2007)
02/07/2007	10	MINUTES ORDER held before Judge Manuel L. Real :, NOTICE VACATING ADR REFERRAL by Judge Manuel L. Real : Notice is given to the parties that the reference to the Alternative Dispute Resolution Pilot Program, General Order 02-07 is vacated. All further settlement procedures in this action shall be pursuant to Local Rule R 16-14.Court Reporter: Not Present. (yc,) (Entered: 02/09/2007)
02/08/2007	12	ORDER by Judge Audrey B. Collins, re APPLICATION of Thomas L Casagrande for Leave to Appear Pro Hac Vice. FEE PAID. 6 (yc,) (Entered: 02/12/2007)
02/08/2007	13	ORDER by Judge Florence-Marie Cooper, re APPLICATION of Shane Nelson for Leave to Appear Pro Hac Vice. FEE PAID. 4 (yc,) (Entered: 02/14/2007)
02/08/2007	14	APPLICATION AND ORDER of Non-Resident Attorney to Appear in a Specific Case by Brian L. Jackson as counsel for Plaintiff Guardian Media Technologies Ltd, designating Jan L. Handzik as local counsel. Fee PAID. Approved by Judge Audrey B. Collins.(yc,) (Entered: 02/14/2007)
02/08/2007	15	APPLICATION AND ORDER of Non-Resident Attorney to Appear in a Specific Case by Michael S. Dowler as counsel for Plaintiff Guardian Media Technologies Ltd, designating Jan L. Handzlik as local counsel. Fee PAID. Approved by Judge Audrey B. Collins.(yc,) (Entered: 02/14/2007)
02/14/2007	17	NOTICE TO COUNSEL by Judge Manuel L. Real, This case has been assinged to the calendar of Judge manuel L. Real. Counsel are advised that the court expects strict compliance with the provisions of the local rules and the Federal Rules of civil procedure. (see file)(yc,) (Entered: 02/15/2007)
03/16/2007	18	PROOF OF SERVICE re Notice to Counsel Pursuant to Paragraph 9 of Notice to Counsel; Exhibit 17 , was served on 3/16/2007 filed by plaintiff Guardian Media Technologies Ltd. (jp) (Entered: 04/02/2007)
04/20/2007		Summons Issued re first Amended Complaint, 19 as to Harsper Co Ltd, Harsper USA Inc, Santax Groups Corp, SVA (Group) Co Ltd, Sva Group (USA)Inc, Syntax-Brillan Corp, LG Electronics Inc, LG Electronics USA Inc. (yc,) (Entered: 04/24/2007)
04/23/2007	19	FIRST AMENDED COMPLAINT against defendants Harsper Co Ltd, Harsper USA Inc, Santax Groups Corp, SVA (Group) Co Ltd, Sva Group (USA)Inc, Syntax-Brillan Corp, LG Electronics Inc, LG Electronics USA Inc amending Complaint - (Discovery) 1 ,filed by plaintiff Guardian Media Technologies Ltd (yc,) (Entered: 04/24/2007)

04/24/2007	21	PROOF OF SERVICE Executed by Plaintiff Guardian Media Technologies Ltd, upon LG Electronics USA Inc served on 4/18/2007, answer due 5/8/2007. The Summons and Complaint were served by Personally service, by not cite statute, upon Catherine Edwards, is described to the best of deponents as follows Sex Femal; white, Blonde, 35 age, Height 5 8 and weight 160-170 pounds. Due Dilligence declaration NOT attached. Original Summons NOT Returned. (et) (Entered: 05/14/2007)
05/07/2007	20	STIPULATION AND ORDER by Judge Manuel L. Real : IT IS HEEBY STIPULATED THAT defendants shall be granted thirty day extension of time in which to answer the complaint, answer which would have been due on 5/7/2007 shall now be due 6/7/2007.(yc,) (Entered: 05/09/2007)
05/30/2007	22	PROOF OF SERVICE Executed upon Syntax-Brillan Corp served on 5/21/2007, answer due 6/10/2007. The Summons and Complaint were served by personally delivering service, by State statute, upon Wendy Chen, authorized agent. Due Dilligence declaration NOT attached. Original Summons NOT attached. (yc) (Entered: 06/04/2007)
05/30/2007	23	PROOF OF SERVICE Executed upon Syntax-Brillian Corporation The Summons and first amended Complaint were served by personally delivering service, by Federal statute, upon Michael Miller- General Counsel. Due Dilligence declaration NOT attached. Original Summons NOT returned. (yc) (Entered: 06/04/2007)
05/30/2007	24	PROOF OF SERVICE Executed upon Harsper USA Inc served on 5/21/2007, answer due 6/10/2007. The Summons and Complaint were served by personally delivering copies service, by State statute, upon Karen Jin, Authorized Agent. Due Dilligence declaration NOT attached. Original Summons returned. (yc) (Entered: 06/05/2007)
06/01/2007	26	STIPULATION for Extension of Time for Defendants Syntax-Brillian corporaiton and Syntax Groups Corporaiton to Respond to First Amended Complaint as to Santax Groups Corp answer now due 7/10/2007; Syntax-Brillan Corp answer now due 7/11/2007, filed by Defendants Santax Groups Corp; Syntax-Brillan Corp.(shb) (Entered: 06/07/2007)
06/04/2007	25	NOTICE OF DISMISSAL filed by plaintiff Guardian Media Technologies Ltd pursuant to FRCP 41a(1) as to LG Electronics Inc, LG Electronics USA Inc. Each party shall bear its own expenses, costs, and fees. Nothing in this dismissal affects Guardian's claims against any of the other named defendants. (yc) (Entered: 06/06/2007)
06/27/2007	27	PROOF OF SERVICE Executed upon Sva Group (USA)Inc served on 6/27/2007, answer due 7/17/2007. The Summons and Complaint were served by personally delivering copies service, by Federal statute, upon Agustin Aguilar, Customer Service Manager. Due Dilligence declaration Not attached. Original Summons NOT returned only copy of summons attached. (yc) (Entered: 07/03/2007)
07/03/2007	30	CERTIFATION AND NOTICE of Interested Parties filed by Defendant SVA (Group) Co Ltd. (yc) (Entered: 07/20/2007)

07/10/2007	28	NOTICE of Interested Parties filed by Defendant Syntax-Brillan Corp AND Syntax Groups Corporation. (yc) (Entered: 07/17/2007)
07/10/2007	29	ANSWER first Amended Complaint, 19 , and COUNTERCLAIM against Guardian Media Technologies Ltd filed by defendants Syntax-Brillan Corp. (yc) (Entered: 07/18/2007)
07/13/2007	31	ANSWER to First Amended Complaint, 19 , COUNTERCLAIM against Guardian Media Technologies Ltd filed by defendant Sva Group (USA)Inc. (rrey) (Entered: 07/24/2007)
08/01/2007	32	Motion for Entry of Default against defendant Harsper USA Inc filed by plaintiff Guardian Media Technologies Ltd.Motion set for hearing on 9/4/2007 at 10:00 AM before Judge Manuel L. Real. (yc) Modified on 8/3/2007 (yc,). (Entered: 08/03/2007)
08/01/2007	33	NOTICE OF MOTION re Motion for Entry of Default against defendant Harsper USA Inc 32 filed by Plaintiff Guardian Media Technologies Ltd. (yc) (Entered: 08/03/2007)
08/01/2007	34	DECLARATION of Michael S. Dowler in support Motion for Entry of Default against defendantHarsper USA Inc 32 filed by Plaintiff Guardian Media Technologies Ltd. (yc) (Entered: 08/03/2007)
08/01/2007	35	PROOF OF SERVICE filed by plaintiff Guardian Media Technologies Ltd, re Motion for Entry of Default against defendantHarsper USA Inc 32 , Declaration (Motion related) 34 , Notice of Motion 33 served on 8/1/2007. (yc) (Entered: 08/03/2007)
08/01/2007	36	ORDER on application of non-resident attorney Stephen L. Lundwall to appear in a specific case. on behalf of plaintiff Guardian Media technologies, Ltd, designation of Jan L. Handzlik.(yc) (Entered: 08/03/2007)
08/01/2007	37	APPLICATION of non resident attorney Stephan L. Lundwall to Appear in a specific case. filed by plaintiff Guardian Media Technologies Ltd. (yc) (Entered: 08/03/2007)
08/01/2007	38	REPLY to Counterclaim 31 OF SVA GROUP (USA). filed by plaintiff Guardian Media Technologies Ltd.(yc) (Entered: 08/03/2007)
08/01/2007	39	REPLY TO Counterclaim 29 of Syntax-Brillian Corporation filed by counterdefendants Guardian Media Technologies Ltd.(yc) (Entered: 08/03/2007)
08/02/2007	40	Notice of Withdrawal of Motion for Entry of Default against defendantHarsper USA Inc 32 filed by plaintiff Guardian Media Technologies Ltd. (yc) (Entered: 08/06/2007)
08/03/2007	41	REQUEST for Clerk to Enter Default against defendants Harsper USA Inc filed by plaintiff Guardian Media Technologies Ltd. (yc) (Entered: 08/08/2007)
08/03/2007	42	DECLARATION of Michael S Dowler in support REQUEST for Clerk to Enter Default against defendants Harsper USA Inc 41 filed by Plaintiff

		Guardian Media Technologies Ltd. (yc) (Entered: 08/08/2007)
08/07/2007	50	DECLARATION of Jan L. Handzlik re: EX PARTE APPLICATION for Leave to to conduct the 26(f) early meeting of counsel via teleconference; declaration of Shane A. Nelson in support 49 filed by Plaintiff Guardian Media Technologies Ltd. (tami) (Entered: 08/24/2007)
08/08/2007	43	ORDER by Judge Manuel L. Real, GRANING Ex part application for leave to conduct the 26(f) early meeting of counsel with answering defendants via teleconference. (see file)(yc) (Entered: 08/10/2007)
08/08/2007	49	EX PARTE APPLICATION for Leave to to conduct the 26(f) early meeting of counsel via teleconference; declaration of Shane A. Nelson in support filed by plaintiff Guardian Media Technologies Ltd. Lodged proposed order.(tami) (Entered: 08/24/2007)
08/08/2007	51	SUPPLEMENTAL FILING re EX PARTE APPLICATION for Leave to to conduct the 26(f) early meeting of counsel via teleconference; declaration of Shane A. Nelson in support 49 filed by Plaintiff Guardian Media Technologies Ltd. (rrey) (Entered: 08/28/2007)
08/13/2007	44	APPLICATION of NON-RESIDENT ATTORNEY JOHN P. POLIAK to Appear in a specific case filed by defendant Syntax-Brillan Corp. Lodged ORDER. (yc) (Entered: 08/16/2007)
08/13/2007	45	APPLICATION of MAXWELL J. PETERSEN for Leave to Appear Pro Hac Vice filed by defendant Syntax-Brillan Corp. Lodged order. (yc) (Entered: 08/16/2007)
08/20/2007	46	ORDER by Judge Manuel L. Real, re APPLICATION of JOHN P. POLIAK for Leave to Appear Pro Hac Vice 44 is granted. (yc) (Entered: 08/22/2007)
08/20/2007	47	ORDER by Judge Manuel L. Real, re APPLICATION of MAXWELL J. PETERSEN for Leave to Appear Pro Hac Vice 45 is granted. (yc) (Entered: 08/22/2007)
08/22/2007	48	JOINT REPORT Rule 26(f) Discovery Plan ; estimated length of trial 4 days, filed by Plaintiff Guardian Media Technologies Ltd, Defendants Sva Group (USA)Inc, Syntax-Brillan Corp. (lra) (Entered: 08/24/2007)
09/07/2007	52	ANSWER to First Amended Complaint, 19 with JURY DEMAND filed by defendant SVA (Group) Co Ltd.(bg) (Entered: 09/11/2007)
09/07/2007		FAX number for Attorney Lisa Sharrock Glasser is 949-760-5200. (bg) (Entered: 09/11/2007)
09/07/2007	53	CERTIFICATION AND NOTICE of Interested Parties filed by Defendant SVA (Group) Co Ltd. (bg) (Entered: 09/13/2007)
09/10/2007	56	NOTICE OF JOINT MOTION AND JOINT MOTION for a Stay pending Reexamination of Patents-in-Suit filed by plaintiff Guardian Media Technologies Ltd, defendants Santax Groups Corp, Sva Group (USA)Inc, Syntax-Brillan Corp.Motion set for hearing on 10/1/2007 at 10:00 AM before Judge Manuel L. Real. (ad) (Entered: 09/17/2007)

09/10/2007	57	MEMORANDUM in Support of JOINT MOTION to Stay Proceedings pending Reexamination 56 filed by Plaintiff Guardian Media Technologies Ltd, Defendants Santax Groups Corp, Sva Group (USA)Inc, Syntax-Brillan Corp. (ad) (Entered: 09/17/2007)
09/10/2007	58	DECLARATION of John P Poliak in support of JOINT MOTION for a Stay pending Reexamination of Patents-in-Suit 56 filed by Plaintiff Guardian Media Technologies Ltd, Defendants Santax Groups Corp, Sva Group (USA) Inc, Syntax-Brillan Corp. (ad) (Entered: 09/17/2007)
09/12/2007	54	NOTICE of Change of Attorney Information: Casondra K Ruga is no longer attorney of record in this case for the reason indicated in the G-06 Notice. Filed by defendants Syntax-Brillan Corp (ak) (Entered: 09/14/2007)
09/14/2007	55	NOTICE of Change of Attorney Information: Brian L Jackson is no longer attorney of record in this case for the reason indicated in the G-06 Notice. Filed by plaintiff Guardian Media Technologies Ltd (ak) (Entered: 09/17/2007)
09/24/2007	59	NOTICE OF JOINDER AND JOINDER OF DEFENDANT AND COUNTERCLAIMANT SVA GROUP (USA), INC. TO MOTION OF DEFENDANT SYNTAX-BRILLIAN CORPORATION FOR A STAY PENDING REEXAMINATION OF PATENTS-IN-SUIT 56 filed by Counter Claimant Sva Group (USA)Inc, Defendant Sva Group (USA)Inc. (ch) (Entered: 09/25/2007)
09/28/2007	60	REQUEST to Substitute attorney Richard H Lee in place of attorney Scott D Baskin and Lisa Sharrock Glasser filed by Defendant SVA (Group) Co Ltd. Lodged proposed order. (shb) (Entered: 09/29/2007)
09/28/2007	61	ORDER by Judge Manuel L. Real GRANTING REQUEST to Substitute attorney Richard H Lee in place of attorney Scott D Baskin and Lisa Sharrock Glasser 60 (shb) (Entered: 10/03/2007)
10/01/2007	62	MINUTES OF Motion Hearing held before Judge Manuel L. Real :, RE: Syntax-Brillian's MOTION to Stay Case pending Reexamination of Patents-in-Suit 56 The Court DENIES the motion. Court Reporter: Sheri Kleeger.(wh) (Entered: 10/03/2007)
10/01/2007	63	MEMORANDUM of Points and Authorities in Support of Plaintiff Guardian Media Technologies Ltd.'s Opposition to Syntax-Brillian Corp.'s and SVA Group (USA), Inc.'s MOTION to Stay 56 filed by Plaintiff Guardian Media Technologies Ltd. (ch) (Entered: 10/04/2007)
10/01/2007	64	DECLARATION of Shane A. Nelson in Support of Plaintiff Guardian Media Technologies Ltd.'s Opposition to Syntax-Brillian Corp.'s and SVA Group USA, Inc.'s MOTION to Stay 56 filed by Plaintiff Guardian Media Technologies Ltd. (ch) (Entered: 10/04/2007)
10/01/2007	65	DECLARATION of Thomas E. Coverstone in Support of Plaintiff Guardian Media Technologies Ltd.'s Opposition to Syntax-Brillian Corp.'s and SVA Group (USA), Inc's MOTION to Stay 56 filed by Plaintiff Guardian Media Technologies Ltd. (ch) (Entered: 10/04/2007)

10/01/2007	66	PROOF OF SERVICE filed by plaintiff Guardian Media Technologies Ltd, re Declaration (Motion related) 64 , Memorandum in Support of Motion, 63 , Declaration (Motion related) 65 served on 10/1/07. (ch) (Entered: 10/04/2007)
10/01/2007	67	JOINDER in MOTION for a Stay pending Reexamination of Patents-in-Suit 56 filed by Defendant SVA (Group) Co Ltd. (jag) (Entered: 10/09/2007)
10/05/2007	68	NOTICE of Change of Attorney Information: Casondra K Ruga is no longer attorney of record in this case for the reason indicated in the G-06 Notice. Filed by defendants Syntax-Brillan Corp (ak) (Entered: 10/10/2007)
10/09/2007	69	MINUTES OF IN CHAMBERS ORDER held before Judge Manuel L. Real : Set Deadlines/Hearings:Discovery cut-off 1/14/2008.,Exhibit List due by 1/14/2008.,Witness List due by 1/14/2008.,Proposed Pretrial Order due by 1/28/2008., Final Pretrial Conference set for 2/4/2008 11:00 AM before Judge Manuel L. Real.,Jury Trial set for 3/4/2008 09:00 AM before Judge Manuel L. Real.(wh) (Entered: 10/10/2007)
10/16/2007	70	STIPULATED PROTECTIVE ORDER by Judge Rosalyn M. Chapman: All documents, electronically stored information, materials, items, things, and/or information produced either by a party or by a third party to any of the parties in this case shall be governed by this Protective Order. IT IS SO ORDERED, as amended at paragraphs 11(c) and 14. (See document for further details.) (pcl) (Entered: 10/16/2007)
10/29/2007	74	NOTICE OF MOTION AND MOTION of Defendants SVA (Group) Co., Ltd. and SVA Group (USA), Inc. to Stay Proceedings in Light of Pending Reexamination of Patent-In-Suit filed by defendant and Counterclaimant Sva Group (USA)Inc.Motion set for hearing on 11/19/2007 at 10:00 AM before Judge Manuel L. Real. (vh) (Entered: 11/05/2007)
10/29/2007	75	MEMORANDUM in Support of Defendants SVA (Group) Co., Ltd. and SVA Group (USA), Inc. to Stay Proceedings in Light of Pending Reexamination of Patent-In-Suit re MOTION to Stay Case pending completion of the U.S. Patent and Trademark Office's patent reexamination of U.S. Patent No. 4,930,160 74 filed by Counter Claimant Sva Group (USA)Inc, Defendant Sva Group (USA)Inc. (vh) (Entered: 11/05/2007)
10/29/2007	76	DECLARATION of Richard H. Lee in Support of Motion of Defendants SVA (Group) Co., Ltd. and SVA Group (USA), Inc. to Stay Proceedings in Light of Pending Reexamination of Patent-In-Suit re MOTION to Stay Case pending completion of the U.S. Patent and Trademark Office's patent reexamination of U.S. Patent No. 4,930,160 74 filed by Counter Claimant Sva Group (USA)Inc, Defendant Sva Group (USA)Inc. (vh) Modified on 11/5/2007 (vh,). (Entered: 11/05/2007)
10/29/2007	90	NOTICE of Change of Attorney Information:, changing e-mail to dc@dchanglaw.com. for attorney David K.W. Chang. Filed by counter claimant Sva Group (USA)Inc (cbr) (Entered: 11/15/2007)
10/31/2007	71	APPLICATION OF NON-RESIDENT ATTORNEY Anthony C Roth for Leave to Appear Pro Hac Vice. FEE PAID. filed by defendant SVA (Group) Co Ltd. Lodged proposed order. (shb) Additional attachment(s) added on

		11/1/2007 (shb,). Modified on 11/1/2007 (shb,). (Entered: 11/01/2007)
11/01/2007	72	NOTICE SYNTAX-BRILLIAN CORPORATION'S NOTICE OF JOINDER AND JOINDER TO MOTION TO STAY PROCEEDINGS BY DEFENDANTS SVA GRIUPS CORP AND SVA (USA) INC filed by DEFENDANT Syntax-Brillan Corp. (Petersen, Maxwell) (Entered: 11/01/2007)
11/01/2007	73	ORDER by Judge Manuel L. Real GRANTING APPLICATION OF NON-RESIDENT ATTORNEY Anthony C Roth for Leave to Appear Pro Hac Vice. FEE PAID. 71 (shb) (Entered: 11/01/2007)
11/05/2007	77	NOTICE TO FILER OF DEFICIENCIES in Electronically Filed Documents. The following deficiency was found Incorrect Document Type selected (i.e., event selected does not match document attached).For the correct event, go to "Responses, Replies and Other Motion Related Documents" and select " Joinder (motion Related)., action was or will be taken by Filer Note: In response to this notice the court may order: 1) an amended or correct document to be filed, 2) the: RE Notice (Other) 72 (pj) (Entered: 11/05/2007)
11/05/2007	78	JOINDER in MOTION to Stay Case pending completion of the U.S. Patent and Trademark Office's patent reexamination of U.S. Patent No. 4,930,160 74 filed by Defendant Syntax-Brillan Corp. (Petersen, Maxwell) (Entered: 11/05/2007)
11/05/2007	79	MEMORANDUM in Opposition to MOTION to Stay Case pending completion of the U.S. Patent and Trademark Office's patent reexamination of U.S. Patent No. 4,930,160 74 filed by Plaintiff Guardian Media Technologies Ltd. (Attachments: # 1 Declaration Thomas E. Coverstone# 2 Declaration Shane A. Nelson# 3 Exhibit 1-6# 4 Exhibit 7-12# 5 Exhibit 13-14# 6 Exhibit 15-16# 7 Exhibit 17# 8 Exhibit 18# 9 Exhibit 19# 10 Exhibit 20# 11 Exhibit 21# 12 Exhibit 22# 13 Exhibit 23# 14 Exhibit 24 (pt 1 of 3)# 15 Exhibit 24 (pt 2 of 3)# 16 Exhibit 24 (pt 3 of 3)# 17 Exhibit 25 (pt 1 of 2)# 18 Exhibit 25 (pt 2 of 2)# 19 Exhibit 26# 20 Exhibit 27 (ot 1 of 2)# 21 Exhibit 27 (pt 2 of 2)# 22 Exhibit 28-30# 23 Exhibit 31-33# 24 Exhibit 34-35# 25 Exhibit 36# 26 Exhibit 37# 27 Exhibit 38 (pt 1 of 2)# 28 Exhibit 38 (pt 2 of 2)# 29 Exhibit 39 (pt 1 of 2)# 30 Exhibit 39 (pt 2 of 2)# 31 Exhibit 40# 32 Exhibit 41-44# 33 Exhibit 45-59# 34 Exhibit 60-62)(Dowler, Michael) (Entered: 11/05/2007)
11/07/2007	80	MINUTES OF IN CHAMBERS ORDER held before Judge Manuel L. Real : re: Defendants' MOTION to Stay Case pending completion of the U.S. Patent and Trademark Office's patent reexamination of U.S. Patent No. 4,930,160 74 is continued for hearing from 11/19/07 to 11/26/07 at 10 am; opposition remains due 11/5/07; reply remains due 11/13/07. (wh) (Entered: 11/07/2007)
11/12/2007	81	NOTICE OF MOTION AND MOTION to Transfer Case to Southern District of California filed by Defendant Syntax-Brillan Corp.Motion set for hearing on 12/3/2007 at 10:00 AM before Judge Manuel L. Real. (Falzone, Leah) (Entered: 11/12/2007)
11/12/2007	82	DECLARATION of Darin Margules In Support of Motion to Transfer MOTION to Transfer Case to Southern District of California 81 filed by Defendant Syntax-Brillan Corp. (Attachments: # 1 Exhibit Exhibit 1# 2

		Exhibit Exhibit 2# 3 Exhibit Exhibit 3# 4 Exhibit Exhibit 4# 5 Exhibit Exhibit 5# 6 Exhibit Exhibit 6)(Falzone, Leah) (Entered: 11/12/2007)
11/13/2007	86	NOTICE OF JOINDER and JOINDER OF DEFENDANT SVA (GROUP) CO., LTD. TO DEFENDANT SYNTAX-BRILLIAN CORPORAITON'S MOTION to Transfer Venue to Southern District of California 81 filed by Defendant SVA (Group) Co Ltd.Motion set for hearing on 12/3/2007 at 10:00 AM before Judge Manuel L. Real. (shb) (Entered: 11/15/2007)
11/13/2007	87	SUPPLEMENT DECLARATION OF RICHARD H LEE FOR REPLY BRIEF OF DEFENDANTS SVA (GROUP) AND SVA GROUP (USA) INC IN SUPPORT OF THEIR MOTION to Stay Proceedings in Light of Pending Reexamination of Patent-In-Suit 74 filed by Defendant SVA (Group) Co Ltd. (shb) (Entered: 11/15/2007)
11/13/2007	88	NOTICE OF ERRATA MOTION FOR MOTION OF Defendants SVA (Group) Co., Ltd. and SVA Group (USA), Inc. to Stay Proceedings in Light of Pending Reexamination of Patent-In-Suit 74 filed by Defendant SVA (Group) Co Ltd. (shb) (Entered: 11/15/2007)
11/13/2007	89	REPLY BRIEF of Defendants SVA (Group) Co, LTd and SVA Group (USA) Inc in support of their MOTION to stay proceedings in light of pending reexamination of patent-in suit 74 filed by Defendant SVA (Group) Co Ltd. (shb) (Entered: 11/15/2007)
11/15/2007	83	PROOF OF SERVICE filed by Defendant Syntax-Brillan Corp, re MOTION to Transfer Case to Southern District of California 81 served on November 12, 2007. (Falzone, Leah) (Entered: 11/15/2007)
11/15/2007	84	PROOF OF SERVICE filed by DEFENDANT Syntax-Brillan Corp, re MOTION to Transfer Case to Southern District of California 81 served on NOVEMBER 12, 2007. (Falzone, Leah) (Entered: 11/15/2007)
11/15/2007	85	PROOF OF SERVICE filed by DEFENDANT Syntax-Brillan Corp, re MOTION to Transfer Case to Southern District of California 81 served on NOVEMBER 12, 2007. (Falzone, Leah) (Entered: 11/15/2007)
11/19/2007	91	MEMORANDUM in Opposition to MOTION to Transfer Case to Southern District of California 81 filed by Plaintiff Guardian Media Technologies Ltd. (Attachments: # 1 Declaration Shane A. Nelson# 2 Exhibit 1- 5# 3 Exhibit 6 - 13)(Dowler, Michael) (Entered: 11/19/2007)
11/26/2007	92	NOTICE OF MOTION AND MOTION to Compel DISCOVERY filed by DEFENDANT Syntax-Brillan Corp.Motion set for hearing on 12/17/2007 at 10:00 AM before Judge Manuel L. Real. (Attachments: # 1 JOINT STIPULATION# 2 DECLARATION OF JOHN POLIAK# 3 EXH. A# 4 EXH. B# 5 EXH. C# 6 EXH. D# 7 EXH. E# 8 EXH. F# 9 EXH. G# 10 EXH. H# 11 EXH. I# 12 EXH. J# 13 EXH. K# 14 EXH. L# 15 EXH. M# 16 EXH. N# 17 EXH. O# 18 DECLARATION OF SHANE NELSON# 19 EXH. P# 20 EXH. Q# 21 EXH. R)(Petersen, Maxwell) (Entered: 11/26/2007)
11/26/2007	93	REPLY in support *of Motion to Transfer to Southern District*, 81 filed by Defendant Syntax-Brillan Corp. (Falzone, Leah) Modified on 11/27/2007

		(shb,). (Entered: 11/26/2007)
11/26/2007	94	DECLARATION of Michael Miller in support of MOTION to Transfer Case to Southern District of California 81 filed by Defendant Syntax-Brillan Corp. (Falzone, Leah) (Entered: 11/26/2007)
11/26/2007	95	DECLARATION of Leah Phillips Falzone in support of MOTION to Transfer Case to Southern District of California 81 filed by Defendant Syntax-Brillan Corp. (Falzone, Leah) (Entered: 11/26/2007)
11/26/2007	96	DECLARATION of Daric Wong in support of MOTION to Transfer Case to Southern District of California 81 filed by Defendant Syntax-Brillan Corp. (Falzone, Leah) (Entered: 11/26/2007)
11/26/2007	97	MINUTES OF Motion Hearing held before Judge Manuel L. Real RE: MOTION to Stay Case pending completion of the U.S. Patent and Trademark Office's patent reexamination of U.S. Patent No. 4,930,160 74 . The Court DENIES the motion. Plaintiff shall submit a proposed order.Court Reporter: Sheri Kleeger. (rrey) (Entered: 11/27/2007)
11/27/2007	98	NOTICE TO FILER OF DEFICIENCIES in Electronically Filed Documents. The following deficiency was found PROOF OF SERVICE is missing for non-electrically served parties, action was or will be taken by - Note: In response to this notice the court may order: 1) an amended or correct document to be filed, 2) the document stricken, or 3) take other action as the court deems appropriate. (shb) (Entered: 11/27/2007)
11/27/2007	99	PROOF OF SERVICE re MOTION to Transfer Case to Southern District of California 81 filed by Defendant Syntax-Brillan Corp. (Falzone, Leah) (Entered: 11/27/2007)
11/28/2007	100	[PROPOSED] ORDER DENYING DEFENDANTS SVA (GROUP) CO., LTD'S AND SVA GROUP (USA), INC.'S MOTION TO STAY PROCEEDINGS PENDING REEXAMINATION re MOTION to Stay Case pending completion of the U.S. Patent and Trademark Office's patent reexamination of U.S. Patent No. 4,930,160 74 filed by Plaintiff Guardian Media Technologies Ltd. (Dowler, Michael) (Entered: 11/28/2007)
11/29/2007	101	NOTICE TO FILER OF DEFICIENCIES in Electronically Filed Documents. The following deficiency was found: Incorrect document event type selected. Should have prepared a Notice of Lodging and separately attached the proposed order. NOTE: In response to this notice the court may order: 1) an amended or correct document to be filed, 2) the document stricken, or 3) take other action as the court deems appropriate.: RE Motion Related Document, 100 (shb) (Entered: 11/29/2007)
11/29/2007	102	ORDER DENYING DEFENDANTS SVA (GROUP) CO., LTD'S AND SVA GROUP (USA) INC.'S MOTION TO STAY PROCEEDINGS PENDING REEXAMINATION by Judge Manuel L. Real DENYING MOTION to Stay Case pending Reexamination of Patents-in-Suit 56 , Joinder (Motion Related) 78 , Notice (Other) 72 , MOTION to Stay Case pending completion of the U.S. Patent and Trademark Office's patent reexamination of U.S. Patent No. 4,930,160 74 , Joinder (Motion Related), Joinder (Motion Related) 59 (shb)

		Modified on 11/30/2007 (shb,). (Entered: 11/30/2007)
11/30/2007	103	MINUTES OF IN CHAMBERS ORDER held before Judge Manuel L. Real : re: MOTION to Transfer Case to Southern District of California 81 . Motion continued to 12/17/2007 at 10:00 AM before Judge Manuel L. Real. (rrey) (Entered: 11/30/2007)
12/04/2007	104	NOTICE OF DISMISSAL filed by Plaintiff Guardian Media Technologies Ltd pursuant to FRCP 41a(1) as to Syntax-Brillan Corp, Santax Groups Corp. (Dowler, Michael) (Entered: 12/04/2007)
12/05/2007	105	DEFAULT BY CLERK ENTERED as to Defendant Harsper USA Inc regarding FIRST AMENDED COMPLAINT (pj) (Entered: 12/05/2007)
12/11/2007	106	MINUTES OF IN CHAMBERS ORDER held before Judge Manuel L. Real : Counsel are notified that on the court's own motion the: Defendant SVA's joinder in motion to transfer to U.S.D.C. So. District of CA is hereby ORDERED CONTINUED FROM DECEMBER 17, 2007 AT 10:00 A.M. TO JANUARY 28, 2008 AT 10:00 A.M. before Judge Manuel L. Real. (bp) (Entered: 12/12/2007)
01/17/2008	107	NOTICE OF DISMISSAL filed by Plaintiff Guardian Media Technologies Ltd pursuant to FRCP 41a(1) as to SVA (Group) Co Ltd, Sva Group (USA) Inc. (Dowler, Michael) (Entered: 01/17/2008)
01/28/2008	108	NOTICE OF DISMISSAL filed by Plaintiff Guardian Media Technologies Ltd pursuant to FRCP 41a(1) as to Harsper Co Ltd, Harsper USA Inc. (Dowler, Michael) (Entered: 01/28/2008)

EXHIBIT G

TO THE DECLARATION OF JEFFREY M. FISHER ISO DEFENDANTS' REPLY ISO MOTION TO DISMISS OR, IN THE ALTERNATIVE, TO TRANSFER VENUE

Acer, Inc. et al v. Technology Properties Limited et al
Case No. 5:08-cv-00877 JF

MMP PATENTS	PENDING IN N.D. CAL.	PENDING IN E.D. TEXAS	ALREADY LITIGATED IN E.D. TEXAS (Tutorial + Markman)
'336 patent	X	X	X
'584 patent	X	X	X
'749 patent	X	X	
'148 patent		X	X
'890 patent		X	

EXHIBIT H

TO THE DECLARATION OF JEFFREY M. FISHER ISO DEFENDANTS' REPLY ISO MOTION TO DISMISS OR, IN THE ALTERNATIVE, TO TRANSFER VENUE

CASREF, CLOSED, JURY, PROTECTIVE-ORDER

U.S. District Court [LIVE]
Eastern District of TEXAS (Marshall)
CIVIL DOCKET FOR CASE #: 2:07-cv-00181-TJW-CE

Acer America Corporation v. Hon Hai Precision Industry Co. Ltd. et al
Assigned to: Judge T. John Ward
Referred to: Magistrate Judge Charles Everingham
Cause: 28:1332 Diversity-Breach of Contract

Date Filed: 05/08/2007
Date Terminated: 07/11/2008
Jury Demand: Both
Nature of Suit: 190 Contract: Other
Jurisdiction: Diversity

Plaintiff

Acer America Corporation
a California corporation

represented by **Harry Lee Gillam, Jr**
Gillam & Smith, LLP
303 South Washington Avenue
Marshall, TX 75670
903-934-8450
Fax: 903-934-9257
Email: gil@gillamsmithlaw.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Alex Verbin Chachkes
Orrick Herrington & Sutcliffe- New York
666 Fifth Avenue
New York, NY 10103-0001
212/506-3748
Fax: 212/506-5151
Email: achachkes@orrick.com
ATTORNEY TO BE NOTICED

Theresa E Norton
Orrick Herrington & Sutcliffe - Menlo Park
1000 Marsh Rd
Menlo Park, CA 94025
650/614-7400
Fax: 650/614-7401
Email: tnorton@orrick.com
ATTORNEY TO BE NOTICED

V.

Defendant

Hon Hai Precision Industry Co. Ltd. *a Taiwanese Corporation* *TERMINATED: 07/07/2008*	represented by	**Daniel Prince** Paul Hastings Janofsky & Walker - Los Angeles 515 S Flower Street 25th Floor Los Angeles, CA 90071-2228 213/683-6169 Fax: 213/627-0705 Email: danielprince@paulhastings.com *TERMINATED: 07/07/2008* **Jay C Chiu** Paul Hastings Janofsky & Walker - Los Angeles 515 S Flower Street 25th Floor Los Angeles, CA 90071-2228 213/683-6315 Fax: 213/627-0705 Email: jaychiu@paulhastings.com *TERMINATED: 07/07/2008* **Katherine F Murray** Paul Hastings Janofsky & Walker - Los Angeles 515 S Flower Street 25th Floor Los Angeles, CA 90071-2228 213/683-6273 Fax: 213/996-3273 Email: katherinemurray@paulhastings.com *TERMINATED: 07/07/2008* **Michael Charles Smith** Siebman Reynolds Burg Phillips & Smith, LLP-Marshall 713 South Washington Marshall, TX 75670 903-938-8900 Fax: 19727674620 Email: michaelsmith@siebman.com *TERMINATED: 08/24/2007* **Todd Snyder** Paul Hastings Janofsky & Walker - Los Angeles 515 S Flower Street 25th Floor Los Angeles, CA 90071-2228

213/683-6000
Fax: 213/627-0705
Email: toddsnyder@paulhastings.com
TERMINATED: 07/07/2008

Vincent K Yip
Paul Hastings Janofsky & Walker - Los Angeles
515 S Flower Street
25th Floor
Los Angeles, CA 90071-2228
213/683-6000
Fax: 213/627-0705
Email: vincentyip@paulhastings.com
TERMINATED: 07/07/2008

Defendant

Quanta Computer Inc.,
a Taiwanese Corporation
TERMINATED: 06/30/2008

represented by **Melvin R Wilcox, III**
Yarbrough - Wilcox, PLLC
100 E. Ferguson, Suite 1015
Tyler, TX 75702
903.595.1133
Fax: 903.595.0191
Email: mrw@yw-lawfirm.com
TERMINATED: 06/30/2008
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Hua Chen
Paul Hastings Janofsky & Walker - Los Angeles
515 S Flower Street
25th Floor
Los Angeles, CA 90071-2228
213/683-6236
Fax: 213/627-0705
Email: huachen@paulhastings.com
TERMINATED: 06/30/2008
ATTORNEY TO BE NOTICED

Katherine F Murray
(See above for address)
TERMINATED: 06/30/2008
ATTORNEY TO BE NOTICED

Michael Charles Smith
(See above for address)
TERMINATED: 08/24/2007

Terrence D Garnett

Paul Hastings Janofsky & Walker - Los Angeles
515 S Flower Street
25th Floor
Los Angeles, CA 90071-2228
213/683-6247
Fax: 213/627-0705
Email: terrygarnett@paulhastings.com
TERMINATED: 06/30/2008
ATTORNEY TO BE NOTICED

Defendant

Wistron Corporation
a Taiwanese Corporation

represented by **Eric M. Albritton**
Attorney at Law
PO Box 2649
Longview, TX 75606
903/757-8449
Fax: 19037587397
Email: ema@emafirm.com
ATTORNEY TO BE NOTICED

Harold H Davis, Jr
Kirkpatrick & Lockhart Preston Gates Ellis - San Francisco
55 2nd Street
Suite 1700
San Francisco, CA 94105
415/882-8200
Fax: 415/882-8202
Email: harold.davis@klgates.com
ATTORNEY TO BE NOTICED

James Scott Hacker
Albritton Law Firm
P. O. Box 2649
Longview, TX 75606
903-757-8449
Fax: 903-758-7397
Email: jsh@emafirm.com
ATTORNEY TO BE NOTICED

Michael Bettinger
Kirkpatrick & Lockhart Preston Gates Ellis - San Francisco
55 2nd Street
Suite 1700
San Francisco, CA 94105
415/882-8200
Fax: 4158828220
Email: mike.bettinger@klgates.com

ATTORNEY TO BE NOTICED

Defendant

Compal Electronics, Inc. *TERMINATED: 06/23/2008*	represented by	**Samuel Franklin Baxter** McKool Smith - Marshall P O Box O 104 East Houston St., Suite 300 Marshall, TX 75670 903/923-9000 Fax: 903-923-9099 Email: sbaxter@mckoolsmith.com *TERMINATED: 06/23/2008* *LEAD ATTORNEY* **Bita Rahebi** Morrison & Foerster LLP - Los Angeles 555 W Fifth St 35th Floor Los Angeles, CA 90013-1024 213/892-5428 Fax: 213/892-5454 Email: brahebi@mofo.com *TERMINATED: 06/23/2008* **Charles S Barquist** Morrison & Foerster LLP - Los Angeles 555 W Fifth St 35th Floor Los Angeles, CA 90013-1024 213/892-5200 Fax: 213/892-5454 Email: cbarquist@mofo.com *TERMINATED: 06/23/2008* **Garret Wesley Chambers** McKool Smith - Dallas 300 Crescent Court Suite 1500 Dallas, TX 75201 214/978-4000 Fax: 12149784044 Email: gchambers@mckoolsmith.com *TERMINATED: 06/23/2008*

Date Filed	#	Docket Text
05/08/2007	1	Acer America Corporation's MOTION to Seal Confidential Exhibits D-J to

		Original Complaint by Acer America Corporation. (Attachments: # 1 Text of Proposed Order)(ch,) (Entered: 05/09/2007)
05/08/2007	2	ORIGINAL COMPLAINT WITH JURY TRIAL DEMAND against Hon Hai Precision Industry Co. Ltd., Quanta Computer Inc.,, Wistron Corporation (Filing fee $ 350.) , filed by Acer America Corporation. (Attachments: # 1 Exhibit A# 2 Exhibit B# 3 Exhibit C# (EXHIBIT'S D - J FILED UNDER SEAL SEE DOCUMENT # 6) (4) Exhibit K# Exhibit L# Exhibit M# Exhibit N)(ch,) Modified on 5/9/2007 (ch,). (Entered: 05/09/2007)
05/08/2007	3	CORPORATE DISCLOSURE STATEMENT filed by Acer America Corporation identifying Acer Inc as Corporate Parent (ch,) Modified on 5/9/2007 (mpv,). (Entered: 05/09/2007)
05/09/2007	4	ORDER REFERRING CASE to Magistrate Judge Charles Everingham. Signed by Judge T. John Ward on 5/8/07. (ch,) (Entered: 05/09/2007)
05/09/2007	5	Magistrate Consent Form Mailed to Acer America Corporation (ch,) Modified on 5/9/2007 (mpv,). Modified on 5/9/2007 (mpv,). (Entered: 05/09/2007)
05/09/2007	6	SEALED ADDITIONAL ATTACHMENTS to Main Document:. (Attachments: # 1 Exhibit D FILED UNDER SEAL# 2 Exhibit E FILED UNDER SEAL# 3 Exhibit F FILED UNDER SEAL# 4 Exhibit G FILED UNDER SEAL# 5 Exhibit H FILED UNDER SEAL# 6 Exhibit I FILED UNDER SEAL# 7 Exhibit J FILED UNDER SEAL)(ch,) (Entered: 05/09/2007)
05/09/2007	7	E-GOV SEALED SUMMONS Issued as to Hon Hai Precision Industry Co. Ltd., Quanta Computer Inc.,, Wistron Corporation. (ch,) (Entered: 05/09/2007)
05/09/2007		Filing fee: $ 350.00, receipt number 2-1-2542 (ch,) (Entered: 05/10/2007)
05/11/2007	8	ORDER granting 1 Motion to Seal exhibits D-J to original complaint. Signed by Judge Charles Everingham on 5/11/07. (ehs,) (Entered: 05/11/2007)
05/22/2007	9	MOTION to Expedite *Plaintiff Acer America Corporation's Expedited Motion for Leave to Take Early Discovery* by Acer America Corporation. (Attachments: # 1 Affidavit Declaration of Harry L. Gillam, Jr.# 2 Exhibit A# 3 Exhibit B# 4 Exhibit C# 5 Exhibit D# 6 Text of Proposed Order)(Gillam, Harry) (Entered: 05/22/2007)
05/24/2007	10	Additional Attachments to Main Document: 9 MOTION to Expedite *Plaintiff Acer America Corporation's Expedited Motion for Leave to Take Early Discovery*.. (Gillam, Harry) (Entered: 05/24/2007)
05/25/2007	11	ANSWER to Complaint with Jury Demand by Quanta Computer Inc.,.(Smith, Michael) (Entered: 05/25/2007)
05/25/2007		Magistrate Consent Form EMailed to Quanta Computer Inc., (sm,) (Entered: 05/25/2007)
05/29/2007	12	MOTION for Extension of Time to File Answer re 2 Complaint, *Unopposed*

		Motion to Extend Time to Respond to the Complaint by Hon Hai Precision Industry Co. Ltd.. (Attachments: # 1 Text of Proposed Order)(Smith, Michael) (Entered: 05/29/2007)
05/29/2007	13	ANSWER to Complaint by Wistron Corporation.(Albritton, Eric) (Entered: 05/29/2007)
05/30/2007		Magistrate Consent Form EMailed to Wistron Corporation (sm,) (Entered: 05/30/2007)
05/31/2007	14	ORDER granting 12 Motion for Extension of Time to Answer. Hon Hai Precision Industry Co. Ltd deadline is extended to 6/12/07 . Signed by Judge Charles Everingham on 5/30/07. (ch,) (Entered: 05/31/2007)
05/31/2007	15	RESPONSE to Motion re 9 MOTION to Expedite *Plaintiff Acer America Corporation's Expedited Motion for Leave to Take Early Discovery filed by Wistron Corporation.* (Albritton, Eric) (Entered: 05/31/2007)
06/04/2007	16	MOTION to Withdraw 9 MOTION to Expedite *Plaintiff Acer America Corporation's Expedited Motion for Leave to Take Early Discovery*, 10 Additional Attachments to Main Document *Plaintiff Acer America Corporation's Unopposed Motion to Withdraw Without Prejudice Acer America Corporation's Expedited Motion for Leave to Take Early Discovery* by Acer America Corporation. (Attachments: # 1 Text of Proposed Order) (Gillam, Harry) (Entered: 06/04/2007)
06/06/2007	17	ORDER granting 16 Motion to Withdraw. Acer America Corporation's Expedited Motion for Leave to take Early Discovery is hereby withdrawn without prejudice . Signed by Judge Charles Everingham on 6/5/07. (ch,) (Entered: 06/06/2007)
06/07/2007	18	APPLICATION to Appear Pro Hac Vice by Attorney Harold H Davis, Jr for Wistron Corporation. (Fee Paid) 1-1-4257 (ch,) (Entered: 06/07/2007)
06/07/2007	19	APPLICATION to Appear Pro Hac Vice by Attorney Michael Bettinger for Wistron Corporation. (Fee Paid) 1-1-4258 (ch,) (Entered: 06/07/2007)
06/11/2007	20	APPLICATION to Appear Pro Hac Vice by Attorney Hua Chen for Quanta Computer Inc.APPROVED (FEE PAID) 2-1-2655 (poa,) (Entered: 06/12/2007)
06/11/2007	21	APPLICATION to Appear Pro Hac Vice by Attorney Terrence D Garnett for Quanta Computer Inc. APPROVED (FEE PAID) 2-1-2656 (poa,) (Entered: 06/12/2007)
06/11/2007	22	APPLICATION to Appear Pro Hac Vice by Attorney Jay C Chiu for Hon Hai Precision Industry Co. Ltd. APPROVED (FEE PAID) 2-1-2657 (poa,) (Entered: 06/12/2007)
06/11/2007	23	APPLICATION to Appear Pro Hac Vice by Attorney Vincent K Yip for Hon Hai Precision Industry Co. Ltd. APPROVED (FEE PAID) 2-1-2658 (poa,) (Entered: 06/12/2007)
06/12/2007	24	MOTION for Extension of Time to File Answer re 2 Complaint, by Hon Hai

		Precision Industry Co. Ltd.. (Attachments: # 1 Text of Proposed Order)(Smith, Michael) (Entered: 06/12/2007)
06/14/2007	25	ORDER granting 24 Motion for Extension of Time to Answer Deadline extended to 6/26/07 . Signed by Judge Charles Everingham on 6/14/07. (ehs,) (Entered: 06/14/2007)
06/14/2007		Answer Due Deadline Updated for Hon Hai Precision Industry Co. Ltd. to 6/26/2007. (ehs,) (Entered: 06/14/2007)
06/15/2007	26	MOTION for Leave to File *Acer America Corporation's Motion for Leave to Amend Complaint* by Acer America Corporation. (Attachments: # 1 Exhibit 1# 2 Text of Proposed Order)(Gillam, Harry) (Entered: 06/15/2007)
06/15/2007	27	MOTION to Seal *Acer America Corporation's Motion to Seal Confidential Exhibits D-J and O-Q to First Amended Complaint* by Acer America Corporation. (Attachments: # 1 Text of Proposed Order)(Gillam, Harry) (Entered: 06/15/2007)
06/19/2007	28	ORDER granting 26 Motion for Leave to Amend Complaint, adding Compal Electronics, Inc.,. Signed by Judge Charles Everingham on 6/18/07. (ch,) (Entered: 06/19/2007)
06/19/2007	29	ORDER granting 27 Motion to Seal Exhibits D-J and O-Q to First Amended Complaint. Signed by Judge Charles Everingham on 6/19/07. (ch,) (Entered: 06/19/2007)
06/20/2007	30	AMENDED COMPLAINT *Acer America Corporation's First Amended Complaint - See original Exhibits A-N attached to Original Complaint* against Compal Electronics, Inc., Hon Hai Precision Industry Co. Ltd., Quanta Computer Inc.,, Wistron Corporation, filed by Acer America Corporation. (Attachments: # 1 Exhibit R)(Gillam, Harry) (Entered: 06/20/2007)
06/20/2007	31	SEALED ADDITIONAL ATTACHMENTS to Main Document: 30 Amended Complaint,. (Attachments: # 1 Exhibit O# 2 Exhibit P# 3 Exhibit Q)(Gillam, Harry) (Entered: 06/20/2007)
06/20/2007	32	E-GOV SEALED SUMMONS Issued as to Compal Electronics, Inc.. (ch,) (Entered: 06/20/2007)
06/28/2007	33	ANSWER to Amended Complaint by Wistron Corporation.(Albritton, Eric) (Entered: 06/28/2007)
06/29/2007	34	MOTION for Extension of Time to File Answer re 30 Amended Complaint, by Hon Hai Precision Industry Co. Ltd.. (Attachments: # 1 Text of Proposed Order)(Smith, Michael) (Entered: 06/29/2007)
06/29/2007	35	MOTION for Extension of Time to File Answer re 30 Amended Complaint, by Quanta Computer Inc.,. (Attachments: # 1 Text of Proposed Order)(Smith, Michael) (Entered: 06/29/2007)
07/05/2007	36	ORDER granting 34 Motion for Extension of Time to Answer. Hon Hai Precision Industry Co. deadline is extended to 7/31/07 . Signed by Judge Charles Everingham on 7/5/07. (ch,) (Entered: 07/05/2007)

07/05/2007		Answer Due Deadline Updated for Hon Hai Precision Industry Co. Ltd. to 7/31/2007. (ch,) (Entered: 07/05/2007)
07/05/2007	37	ORDER granting 35 Motion for Extension of Time to Answer. Quanta's deadline is extended to 7/31/07 . Signed by Judge Charles Everingham on 7/5/07. (ch,) (Entered: 07/05/2007)
07/05/2007		Answer Due Deadline Updated for Quanta Computer Inc., to 7/31/2007. (ch,) (Entered: 07/05/2007)
07/10/2007	38	ANSWER to Amended Complaint by Compal Electronics, Inc..(Baxter, Samuel) (Entered: 07/10/2007)
07/11/2007		Magistrate Consent Form eMailed to Compal Electronics, Inc. (mpv,) (Entered: 07/11/2007)
07/12/2007	39	APPLICATION to Appear Pro Hac Vice by Attorney Charles S Barquist for Compal Electronics, Inc.. (FEE PAID) 2-1-2773 (ehs,) (Entered: 07/12/2007)
07/12/2007	40	APPLICATION to Appear Pro Hac Vice by Attorney Bita Rahebi for Compal Electronics, Inc. (APPROVED)(FEE PAID) 2-1-2773. (ch,) (Entered: 07/13/2007)
07/31/2007	41	*****FILED IN ERROR; PER ATTY; PLEASE IGNORE; REPLACED BY # 43 ***** MOTION for Extension of Time to File Answer re 30 Amended Complaint, by Hon Hai Precision Industry Co. Ltd.. (Attachments: # 1 Text of Proposed Order)(Smith, Michael) Modified on 8/1/2007 (mpv,). (Entered: 07/31/2007)
07/31/2007	42	*****FILED IN ERROR; PER ATTY; PLEASE IGNORE; REPLACED BY # 44 ***** MOTION for Extension of Time to File Answer re 30 Amended Complaint, by Quanta Computer Inc.,. (Attachments: # 1 Text of Proposed Order)(Smith, Michael) Modified on 8/1/2007 (mpv,). (Entered: 07/31/2007)
07/31/2007	43	MOTION for Extension of Time to File Answer re 30 Amended Complaint, *(Replaces docket number 41)* by Hon Hai Precision Industry Co. Ltd.. (Attachments: # 1 Text of Proposed Order)(Smith, Michael) (Entered: 07/31/2007)
07/31/2007	44	MOTION for Extension of Time to File Answer re 30 Amended Complaint, *(Replaces docket number 42)* by Quanta Computer Inc.,. (Attachments: # 1 Text of Proposed Order)(Smith, Michael) (Entered: 07/31/2007)
07/31/2007		*****FILED IN ERROR. Document # 41 and 42, MOTION for Extension of Time to File Answer re 30 Amended Complaint, by Hon Hai Precision Industry Co. Ltd; MOTION for Extension of Time to File Answer re 30 Amended Complaint, by Quanta Computer Inc. PLEASE IGNORE. Documents replaced by #'s 43 and 44 ***** (mpv,) (Entered: 08/01/2007)

08/01/2007	45	ORDER granting 44 Motion for Extension of Time to Answer. Dft Quanta Computer Inc. Deadline is extended to 8/31/07 . Signed by Judge Charles Everingham on 8/1/07. (ch,) (Entered: 08/01/2007)
08/01/2007	46	ORDER granting 43 Motion for Extension of Time to Answer. Answer deadline reset to 8/31/07 for deft Hon Hai Precision Industry . Signed by Judge Charles Everingham on 8/1/07. (ehs,) (Entered: 08/01/2007)
08/01/2007	47	Answer Due Deadline Updated for Quanta Computer Inc., to 8/31/2007. (ch,) (Entered: 08/01/2007)
08/01/2007		Answer Due Deadline Updated for Hon Hai Precision Industry Co. Ltd. to 8/31/2007. (ehs,) (Entered: 08/01/2007)
08/07/2007	48	*****FILED IN ERROR, PER ATTY, PLEASE IGNORE.***** MOTION to Seal by Quanta Computer Inc.,. (Attachments: # 1 Text of Proposed Order granting Quanta Computer Inc.'s motion to seal)(Chen, Hua) Modified on 8/8/2007 (sm,). (Entered: 08/07/2007)
08/07/2007	49	*****FILED IN ERROR, PLEASE IGNORE.***** MOTION to Seal by Hon Hai Precision Industry Co. Ltd.. (Attachments: # 1 Text of Proposed Order granting Hon Hai Precision Industry Co. Ltd.'s motion to seal)(Yip, Vincent) Modified on 8/8/2007 (sm,). (Entered: 08/07/2007)
08/08/2007		*****FILED IN ERROR, PER ATTY, ATTY WILL REFILE LATER***** . Document # 48 and #49, Motions to seal. PLEASE IGNORE.*** (sm,) (Entered: 08/08/2007)
08/09/2007	50	MOTION to Seal by Hon Hai Precision Industry Co. Ltd.. (Attachments: # 1 Text of Proposed Order granting Hon Hai Precision Industry Co. Ltd's Motion to Seal)(Yip, Vincent) (Entered: 08/09/2007)
08/09/2007	51	SEALED MOTION *to Dismiss* by Hon Hai Precision Industry Co. Ltd.. (Attachments: # 1 Affidavit of Ming-Li Lien in support of Hon Hai's Motion to Dismiss# 2 Affidavit of Vincent K. Yip in support of Hon Hai's Motion to Dismiss# 3 Text of Proposed Order granting Hon Hai Precision Industry Co., Ltd.'s Motion to Dismiss)(Yip, Vincent) (Entered: 08/09/2007)
08/10/2007	52	ORDER granting 50 Motion to Seal Hon Hai's motion to dismiss and supporting declarations. Signed by Judge Charles Everingham on 8/10/07. (ehs,) (Entered: 08/10/2007)
08/22/2007	53	MOTION to Withdraw as Attorney *Unopposed* by Hon Hai Precision Industry Co. Ltd.. (Attachments: # 1 Text of Proposed Order)(Smith, Michael) (Entered: 08/22/2007)
08/22/2007	54	MOTION to Withdraw as Attorney *Unopposed* by Quanta Computer Inc.,. (Attachments: # 1 Text of Proposed Order)(Smith, Michael) (Entered: 08/22/2007)

08/24/2007	55	ORDER granting 53 Motion to Withdraw as Attorney. Attorney Michael Charles Smith terminated . Signed by Judge Charles Everingham on 8/24/07. (poa,) (Entered: 08/24/2007)
08/24/2007	56	ORDER granting 54 Motion to Withdraw as Attorney. Attorney Michael Charles Smith terminated . Signed by Judge Charles Everingham on 8/24/07. (poa,) (Entered: 08/24/2007)
08/27/2007	57	MOTION to Seal Document *Acer America Corporation's Unopposed Motion to Seal Confidential Documents* by Acer America Corporation. (Attachments: # 1 Text of Proposed Order)(Gillam, Harry) (Entered: 08/27/2007)
08/27/2007	58	SEALED RESPONSE to Motion re 51 SEALED MOTION *to Dismiss* filed by Acer America Corporation. (Attachments: # 1 Affidavit Declaration of Theresa E. Norton# 2 Exhibit A# 3 Exhibit B# 4 Exhibit C# 5 Exhibit D# 6 Exhibit E# 7 Exhibit F)(Gillam, Harry) (Entered: 08/27/2007)
08/31/2007	59	ORDER granting 57 Motion to Seal Document Acers Opposition to Hon Hai Precision Industry Co., Ltd's Motion to Dismiss and Declarations of Theresa E. Norton in Support of Acer America Oppoisition to Hon Hai's Motion to Dismiss . Signed by Judge Charles Everingham on 8/31/07. (ch,) (Entered: 08/31/2007)
08/31/2007	60	*Defendant's, Quanta Computer, Inc.,* ANSWER to Amended Complaint *(Acer America Corporation First Amended Complaint)* by Quanta Computer Inc.,. (Wilcox, Melvin) (Entered: 08/31/2007)
09/05/2007	61	MOTION to Seal *Hon Hai's Reply to Acer America Corporation's Opposition* by Hon Hai Precision Industry Co. Ltd.. (Attachments: # 1 Text of Proposed Order)(Yip, Vincent) (Entered: 09/05/2007)
09/05/2007	62	SEALED REPLY to Acer America Corporation's Opposition to Hon Hai's Motion (re 51 SEALED MOTION *to Dismiss*) filed by Hon Hai Precision Industry Co. Ltd.. (Yip, Vincent) (Entered: 09/05/2007)
09/06/2007	63	ORDER granting 61 Motion to Seal Hon Hai's Reply Motion. Signed by Judge Charles Everingham on 9/6/07. (ch,) (Entered: 09/06/2007)
09/14/2007	64	CORPORATE DISCLOSURE STATEMENT filed by Hon Hai Precision Industry Co. Ltd. identifying None as Corporate Parent. (Wilcox, Melvin) Modified on 9/14/2007 (mpv,). (Entered: 09/14/2007)
09/14/2007	65	CORPORATE DISCLOSURE STATEMENT filed by Quanta Computer Inc., identifying None as Corporate Parent. (Wilcox, Melvin) Modified on 9/14/2007 (mpv,). (Entered: 09/14/2007)
09/17/2007	66	SUR-REPLY to Reply to Response to Motion re 51 SEALED MOTION *to Dismiss Acer America Corporation's Sur-Reply to Hon Hai's Motion to Dismiss filed by Acer America Corporation.* (Gillam, Harry) (Entered: 09/17/2007)
09/20/2007	67	CORPORATE DISCLOSURE STATEMENT filed by Compal Electronics, Inc. identifying None as Corporate Parent. (Baxter, Samuel) (Entered: 09/20/2007)

09/20/2007	68	NOTICE of Attorney Appearance by Garret Wesley Chambers on behalf of Compal Electronics, Inc. (Chambers, Garret) (Entered: 09/20/2007)
12/21/2007	69	APPLICATION to Appear Pro Hac Vice by Attorney Katherine Murray for Hon Hai Precision Industry Co. Ltd. and Quanta Computer Inc. (FEE PAID) 2-1-3343 (ehs,) (Entered: 12/27/2007)
12/26/2007	70	APPLICATION to Appear Pro Hac Vice by Attorney Todd Snyder for Hon Hai Precision Industry Co. Ltd. (FEE PAID) 2-1-3345 (ehs,) (Entered: 12/27/2007)
03/31/2008	71	ORDER denying 51 Motion to dismiss. Signed by Judge T. John Ward on 3/31/08. (ehs,) (Entered: 03/31/2008)
04/11/2008	72	NOTICE of Hearing: Scheduling Conference set for 5/13/2008, 10:15 AM, in Mag Ctrm (Marshall) before Magistrate Judge Charles Everingham.(delat) (Entered: 04/11/2008)
04/14/2008	73	ANSWER to Amended Complaint by Hon Hai Precision Industry Co. Ltd.. (Murray, Katherine) (Entered: 04/14/2008)
04/17/2008	74	Notice of Scheduling Conference, Proposed Deadlines for Docket Control Order and Discovery Order. Scheduling Conference set for 5/13/2008 10:15 AM before Judge T. John Ward.. Signed by Judge Charles Everingham on 4/17/08. (ch,) (Entered: 04/17/2008)
04/28/2008	75	APPLICATION to Appear Pro Hac Vice by Attorney Alex Verbin Chachkes for Acer America Corporation. (FEE PAID) 2-1-3748(ehs,) (Entered: 04/28/2008)
04/30/2008	76	APPLICATION to Appear Pro Hac Vice by Attorney Theresa E Norton for Acer America Corporation. (APPROVED)(FEE PAID) 2-1-3754 (ch,) (Entered: 05/08/2008)
05/08/2008	77	MOTION to Stay by Hon Hai Precision Industry Co. Ltd.. (Murray, Katherine) Additional attachment(s) added on 5/12/2008 (sm,). (Entered: 05/08/2008)
05/08/2008	78	AFFIDAVIT in Support re 77 MOTION to Stay *filed by Hon Hai Precision Industry Co. Ltd.*. (Murray, Katherine) (Entered: 05/08/2008)
05/08/2008	79	*****FILED IN ERROR, PLEASE IGNORE.***** Additional Attachments to Main Document: 77 MOTION to Stay.. (Murray, Katherine) Modified on 5/12/2008 (sm,). (Entered: 05/08/2008)
05/12/2008	80	NOTICE of Attorney Appearance by James Scott Hacker on behalf of Wistron Corporation (Hacker, James) (Entered: 05/12/2008)
05/12/2008		*****FILED IN ERROR, IS A PROPOSED ORDER TO #77 SO MUST BE ATTACHED TO MOTION (CLERK IS GOING TO ATTACH ORDER TO #77). Document # #79, Additional Attachments. PLEASE IGNORE.*****

		(sm,) (Entered: 05/12/2008)
05/12/2008	81	NOTICE by Quanta Computer Inc., re 77 MOTION to Stay *Notice of Joinder in Motion* (Wied, Peter) Additional attachment(s) Certificate of Service added on 5/14/2008 (mpv,). Modified on 5/14/2008 (mpv,). (Entered: 05/12/2008)
05/12/2008	82	REPORT of Rule 26(f) Planning Meeting. (Attachments: # 1 Exhibit A) (Davis, Harold) (Entered: 05/12/2008)
05/13/2008	83	Minute Entry for proceedings held before Judge Charles Everingham : Scheduling Conference held on 5/13/2008. (jml,) (Entered: 05/14/2008)
05/20/2008	84	TRANSCRIPT of Proceedings held on 5/13/08 Scheduling Conference before Judge Chad Everingham. Court Reporter/Transcriber: Susan Simmons,Telephone number: 903/935-3868. **NOTICE RE REDACTION OF TRANSCRIPTS: The parties have five (5) business days to file with the Court a Notice of Intent to Request Redaction of this transcript. If no such Notice is filed, the transcript will be made remotely electronically available to the public without redaction after 90 calendar days. The policy is located on our website at www.txed.uscourts.gov** Transcript may be viewed at the court public terminal or purchased through the Court Reporter/Transcriber before the deadline for Release of Transcript Restriction. After that date it may be obtained through PACER.. Redaction Request due 6/10/2008. Redacted Transcript Deadline set for 6/20/2008. Release of Transcript Restriction set for 8/18/2008. (lss,) (Entered: 05/20/2008)
05/20/2008	85	STIPULATED PROTECTIVE ORDER. Signed by Judge Magistrate Judge Charles Everingham on 5/20/08. (ehs,) (Entered: 05/20/2008)
05/23/2008	86	*****FILED IN ERROR; PER ATTY; PLEASE IGNORE***** MOTION to Seal Document *Acer America Corporation's Unopposed Motion to Seal Confidential Documents* by Acer America Corporation. (Attachments: # 1 Text of Proposed Order)(Gillam, Harry) Modified on 5/23/2008 (mpv,). (Entered: 05/23/2008)
05/23/2008		*****FILED IN ERROR. Document # 86, Motion to Seal. PLEASE IGNORE. Per Attorney, wrong document attached, attorney will refile correct document.***** (mpv,) (Entered: 05/23/2008)
05/23/2008	87	*****REPLACES #86***** MOTION to Seal Document *Acer America Corporation's Unopposed Motion to Seal Confidential Documents* by Acer America Corporation. (Attachments: # 1 Text of Proposed Order)(Gillam, Harry) Modified on 5/27/2008 (mpv,). (Entered: 05/23/2008)

05/23/2008	88	SEALED RESPONSE IN OPPOSITION *Acer's Opposition to 77 Hon Hai's Motion to Stay Proceedings* by Acer America Corporation. (Attachments: # 1 Declaration of Theresa Norton, # 2 Exhibit 1 Part 1, # 3 Exhibit 1 Part 2, # 4 Exhibit 1 Part 3, # 5 Exhibit 1 Part 3, # 6 Exhibit 1 Part 5, # 7 Exhibit 1 Part 6, # 8 Exhibit 2, # 9 Exhibit 3 Part 1, # 10 Exhibit 3 Part 2, # 11 Exhibit 4 Part 1, # 12 Exhibit 4 Part 2, # 13 Exhibit 4 Part 3, # 14 Exhibit 4 Part 4, # 15 Exhibit 4 Part 5, # 16 Exhibit 4 Part 6, # 17 Exhibit 5 Part 1, # 18 Exhibit 5 Part 2, # 19 Exhibit 5 Part 3, # 20 Exhibit 6, # 21 Exhibit 7 Part 1, # 22 Exhibit 7 Part 2, # 23 Exhibit 7 Part 3, # 24 Text of Proposed Order Denying Hon Hai's Motion to Stay)(Gillam, Harry) Modified on 5/27/2008 (mpv,). (Entered: 05/23/2008)
05/27/2008		NOTICE re 88 SEALED MOTION *Acer's Opposition to Hon Hai's Motion to Stay Proceedings* Docket text has been edited by clerk to correct docket entry. Document is not a motion. Corrected by clerk to read: SEALED RESPONSE IN OPPOSITION Acer's Opposition to 77 Hon Hai's Motion to Stay Proceedings by Acer America Corporation. (mpv,) (Entered: 05/27/2008)
05/28/2008	89	MOTION for Extension of Time to File *Unopposed Motion to Extend Time to File Opposition to Hon Hai's Motion to Stay* by Acer America Corporation. (Attachments: # 1 Text of Proposed Order)(Gillam, Harry) (Entered: 05/28/2008)
05/29/2008	90	ORDER granting 89 Motion for Extension of Time to File Opposition to Hon Hai's motion to stay. Signed by Magistrate Judge Charles Everingham on 5/29/08. (ehs,) (Entered: 05/29/2008)
06/04/2008	91	AMENDED DISCOVERY ORDER. Signed by Magistrate Judge Charles Everingham on 6/4/08. (ehs,) (Entered: 06/04/2008)
06/04/2008	92	DOCKET CONTROL ORDER - Amended Pleadings due by 8/25/2008. Expert Witness List due by 8/4/2008. Joinder of Parties due by 6/12/2008. Motions due by 1/19/2009. Proposed Pretrial Order due by 1/19/2009. Jury Selection set for 2/2/2009 09:00AM before Judge T. John Ward. Pretrial Conference set for 1/26/2009 02:30 PM before Judge T. John Ward. Signed by Magistrate Judge Charles Everingham on 6/4/08. (ehs,) (Entered: 06/04/2008)
06/05/2008	93	REPLY to Response to Motion re 77 MOTION to Stay *Proceedings filed by Hon Hai Precision Industry Co. Ltd.*. (Prince, Daniel) (Entered: 06/05/2008)
06/11/2008	94	Joint MOTION for Extension of Time to File *Joint Motion to Extend Deadline for Initial Disclosures* by Acer America Corporation, Hon Hai Precision Industry Co. Ltd., Quanta Computer Inc.,, Wistron Corporation, Compal Electronics, Inc.. (Attachments: # 1 Text of Proposed Order)(Gillam, Harry) (Entered: 06/11/2008)
06/19/2008	95	STIPULATION of Dismissal *between Acer and Compal* by Acer America Corporation. (Attachments: # 1 Text of Proposed Order)(Norton, Theresa) (Entered: 06/19/2008)
06/19/2008	96	STIPULATION of Dismissal *between Acer and Wistron* by Acer America Corporation. (Attachments: # 1 Text of Proposed Order)(Norton, Theresa) (Entered: 06/19/2008)

06/23/2008	97	ORDER - granting 95 Joint Stipulation of Dismissal. All claims asserted between Acer America Corporaiton and Compal Electronics, Inc. are dismissed without prejudice. Each parties will bear its attorney fee's and costs and other expenses incurred. Signed by Judge T. John Ward on 6/20/08. (ch,) (Entered: 06/23/2008)
06/25/2008	98	STIPULATION of Dismissal *between Acer and Quanta* by Acer America Corporation. (Attachments: # 1 Text of Proposed Order)(Norton, Theresa) (Entered: 06/25/2008)
06/30/2008	99	ORDER - granting 98 Joint Stipulation of Dismissal. All asserted claims against Quanta are dismissed without prejudice. Each parties will bear its own attorney fees costs and other expenses incurred. Signed by Judge T. John Ward on 6/30/08. (ch,) (Entered: 06/30/2008)
07/01/2008	100	STIPULATION of Dismissal *between Acer and Hon Hai* by Acer America Corporation. (Attachments: # 1 Text of Proposed Order)(Norton, Theresa) (Entered: 07/01/2008)
07/07/2008	101	ORDER - granting 100 Joint Stipulation to Dismiss all of Acers Claims Against Hon Hai without prejudice. All asserted claims are dismissed without prejudice. Each parties will bear its own attorney fees, costs and other expenses incurred. Signed by Judge T. John Ward on 7/7/08. (ch,) (Entered: 07/07/2008)
07/11/2008	102	ORDER - granting 96 Joint Stipulation of Dismissal. All asserted claims between parties are hereby dismissed without prejudice. Each parties will bear its own attorney fees, costs and other expenses incurred. Signed by Judge T. John Ward on 7/11/08. (ch,) (Entered: 07/11/2008)

PACER Service Center			
Transaction Receipt			
07/16/2008 17:10:05			
PACER Login:	fb0051	**Client Code:**	23129-tpl
Description:	Docket Report	**Search Criteria:**	2:07-cv-00181-TJW-CE
Billable Pages:	9	**Cost:**	0.72

EXHIBIT I

TO THE DECLARATION OF JEFFREY M. FISHER ISO DEFENDANTS' REPLY ISO MOTION TO DISMISS OR, IN THE ALTERNATIVE, TO TRANSFER VENUE

CLOSED

U.S. District Court
Western District of Wisconsin (Madison)
CIVIL DOCKET FOR CASE #: 3:07-cv-00620-bbc

Acer, Inc. et al v. HEWLETT-PACKARD COMPANY
Assigned to: Chief Judge Barbara B Crabb
Referred to: Magistrate Judge Stephen L Crocker
Cause: 35:271 Patent Infringement

Date Filed: 10/30/2007
Date Terminated: 06/09/2008
Jury Demand: Both
Nature of Suit: 830 Patent
Jurisdiction: Federal Question

Plaintiff

Acer, Inc.

represented by

Connie Merriett
Dechert LLP
2440 W. El Camino Real
Suite 700
Mountain View, CA 94040-1499
(650) 813-4800
Fax: (650) 813-4848
Email: connie.merriett@dechert.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Craig Y Allison
Dechert LLP
2440 W. El Camino Real
Suite 700
Mountain View, CA 94040
650-813-4810
Fax: 650-813-4848
Email: craig.allison@dechert.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

George W Webb, III
Dechert LLP
300 West 6th Street Suite 1850
Austin, TX 78701
512-394-3016
Fax: 512-394-3001
Email: george.webb@dechert.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jacob Gantz
Dechert LLP
Cira Centre
2929 Arch Street

Philadelphia, PA 19104
215-994-2790
Fax: 215-655-2790
Email: jay.gantz@dechert.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John D. van Loben Sels
Dechert, LLP
2440 W. El Camino Real
Suite 700
Mountain View, CA 94040-1499
650-813-4800
Fax: 650-813-4848
Email: john.vanlobensels@dechert.com

LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jonathan D. Baker
Dechert, LLP
2440 W. El Camino Real
Suite 700
Mountain View, CA 94040-1499
650-813-4800
Fax: 650-813-4848
Email: jonathan.baker@dechert.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Justin F. Boyce
Dechert LLP
2440 W. El Camino Real
Suite 700
Mountain View, CA 94040-1499
(650) 813-4800
Fax: (650) 813-4848
Email: justin.boyce@dechert.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Michael J. Modl
Axley Brynelson, LLP
P.O. Box 1767
Madison, WI 53701
608-257-5661
Fax: 608-257-5444
Email: mmodl@axley.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Robert D. Rhoad
Dechert LLP
Princeton Pike Corporate Center
P.O. Box 5218
Princeton, NJ 08543
609-620-3200
Fax: 609-620-3259
Email: robert.rhoad@dechert.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Steven R. Daniels
Dechert LLP
300 W. 6th Street
Suite 1850
Austin, TX 78701
(512) 394-3000
Fax: (512) 394-3001
Email: steven.daniels@dechert.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Valerie Margo Wagner
Dechert, LLP
2440 W. El Camino Real
Suite 700
Mountain View, CA 94040-1499
650-813-4800x4876
Fax: 650-813-4848
Email: valerie.wagner@dechert.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

W. Bryan Farney
Dechert LLP
300 West 6th Street Suite 1850
Austin, TX 78701
512-394-3000
Fax: 512-394-3001
Email: bryan.farney@dechert.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Andrew Clarkowski
Axley Brynelson
2 E. Mifflin St., Ste. 200
Madison, WI 53703
608-283-6705
Fax: 608-257-5444

Email: aclarkowski@axley.com
ATTORNEY TO BE NOTICED

Plaintiff

Acer America Corporation

represented by **Connie Merriett**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Craig Y Allison
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

George W Webb, III
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jacob Gantz
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John D. van Loben Sels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jonathan D. Baker
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Justin F. Boyce
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Michael J. Modl
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Robert D. Rhoad
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Steven R. Daniels

(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Valerie Margo Wagner
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

W. Bryan Farney
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Andrew Clarkowski
(See above for address)
ATTORNEY TO BE NOTICED

V.

Defendant

HEWLETT-PACKARD COMPANY represented by **Brian M. Fogarty**
DLA Piper US LLP
401 B Street
Suite 1700
San Diego, CA 92101
619-699-2620
Fax: 619-699-2701
Email: brian.fogarty@dlapiper.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Brooke A. Beros
DLA Piper US LLP
401 B Street
Suite 1700
San Diego, CA 92101
619-699-2648
Fax: 619-699-2701
Email: brooke.beros@dlapiper.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

David L. Alberti
DLA PIPER US LLP
2000 University Avenue
East Palo Alto, CA 94303
650-833-2052
Fax: 650-687-1157
Email: david.alberti@dlapiper.com

LEAD ATTORNEY
ATTORNEY TO BE NOTICED

David K. Wiggins
DLA Piper US LLP
401 B Street
Suite 1700
San Diego, CA 92101
619-699-2700x2815
Fax: 619-699-2701
Email: david.wiggins@dlapiper.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Edward H. Sikorski
DLA Piper US LLP
401 B Street
Suite 1700
San Diego, CA 92101
619-699-2645
Fax: 619-699-2701
Email: ed.sikorski@dlapiper.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

James Donald Peterson
Godfrey & Kahn, S.C.
One East Main Street, Suite 500
P.O. Box 2719
Madison, WI 53701
608-257-3911
Fax: 608-257-0609
Email: jpeterson@gklaw.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John Allcock
DLA Piper US LLP
401 B Street
Suite 1700
San Diego, CA 92101-4297
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Kathryn Bridget Riley
DLA Piper US LLP
401 B Street
Suite 1700
San Diego, CA 92101
619-699-2842

Fax: 619-699-2701
Email: kathryn.riley@dlapiper.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Mark D. Fowler
DLA PIPER US LLP
2000 University Avenue
East Palo Alto, CA 94303
650-833-2000x2048
Fax: 650-833-2001
Email: mark.fowler@dlapiper.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Nikki Wyll
DLA Piper US LLP
401 B Street
Suite 1700
San Diego, CA 92101
619-699-2879
Email: nikki.wyll@dlapiper.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Richard Thomas Mulloy
DLA Piper US LLP
401 B Street
Suite 1700
San Diego, CA 92101
619-699-4787
Fax: 619-699-2701
Email: richard.mulloy@dlapiper.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Sal Lim
DLA Piper
2000 University Avenue
East Palo Alto, CA 94303
650-833-2101
Fax: 650-833-2001
Email: sal.lim@dlapiper.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Sean C. Cunningham
DLA Piper US LLP
401 B Street
Suite 1700

San Diego, CA 92101
619-699-2700
Email: sean.cunningham@dlapiper.com

LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Counter Claimant

HEWLETT-PACKARD COMPANY represented by **Brian M. Fogarty**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Brooke A. Beros
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

David L. Alberti
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

David K. Wiggins
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Edward H. Sikorski
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

James Donald Peterson
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John Allcock
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Kathryn Bridget Riley
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Mark D. Fowler
(See above for address)

LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Nikki Wyll
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Richard Thomas Mulloy
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Sal Lim
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Sean C. Cunningham
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

V.

Counter Defendant

Acer, Inc. represented by **Connie Merriett**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Craig Y Allison
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

George W Webb, III
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jacob Gantz
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John D. van Loben Sels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jonathan D. Baker
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Justin F. Boyce
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Michael J. Modl
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Robert D. Rhoad
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Steven R. Daniels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Valerie Margo Wagner
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

W. Bryan Farney
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Andrew Clarkowski
(See above for address)
ATTORNEY TO BE NOTICED

Counter Defendant

Acer America Corporation

represented by **Connie Merriett**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Craig Y Allison
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

George W Webb, III
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jacob Gantz
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John D. van Loben Sels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jonathan D. Baker
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Justin F. Boyce
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Michael J. Modl
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Robert D. Rhoad
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Steven R. Daniels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Valerie Margo Wagner
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

W. Bryan Farney
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Andrew Clarkowski
(See above for address)
ATTORNEY TO BE NOTICED

Counter Claimant

Acer, Inc.

represented by **Connie Merriett**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Craig Y Allison
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

George W Webb, III
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jacob Gantz
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John D. van Loben Sels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jonathan D. Baker
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Justin F. Boyce
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Michael J. Modl
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Robert D. Rhoad
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Steven R. Daniels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Valerie Margo Wagner
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

W. Bryan Farney
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Andrew Clarkowski
(See above for address)
ATTORNEY TO BE NOTICED

Counter Claimant

Acer America Corporation represented by **Connie Merriett**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Craig Y Allison
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

George W Webb, III
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jacob Gantz
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John D. van Loben Sels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jonathan D. Baker
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Justin F. Boyce
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Michael J. Modl
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Robert D. Rhoad
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Steven R. Daniels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Valerie Margo Wagner
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

W. Bryan Farney
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Andrew Clarkowski
(See above for address)
ATTORNEY TO BE NOTICED

V.

Counter Defendant

HEWLETT-PACKARD COMPANY represented by **Brian M. Fogarty**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Brooke A. Beros
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

David L. Alberti
(See above for address)

LEAD ATTORNEY
ATTORNEY TO BE NOTICED

David K. Wiggins
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Edward H. Sikorski
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

James Donald Peterson
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John Allcock
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Kathryn Bridget Riley
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Mark D. Fowler
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Nikki Wyll
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Richard Thomas Mulloy
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Sal Lim
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Sean C. Cunningham
(See above for address)

LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Counter Claimant

HEWLETT-PACKARD COMPANY

V.

Counter Defendant

Acer, Inc. represented by **Connie Merriett**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Justin F. Boyce
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Steven R. Daniels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Andrew Clarkowski
(See above for address)
ATTORNEY TO BE NOTICED

Counter Defendant

Acer America Corporation represented by **Connie Merriett**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Justin F. Boyce
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Steven R. Daniels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Andrew Clarkowski
(See above for address)
ATTORNEY TO BE NOTICED

Counter Claimant

Acer, Inc. represented by **Connie Merriett**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Craig Y Allison
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

George W Webb, III
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jacob Gantz
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John D. van Loben Sels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jonathan D. Baker
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Justin F. Boyce
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Michael J. Modl
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Robert D. Rhoad
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Steven R. Daniels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Valerie Margo Wagner
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

W. Bryan Farney
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Andrew Clarkowski
(See above for address)
ATTORNEY TO BE NOTICED

Counter Claimant

Acer America Corporation represented by **Connie Merriett**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Craig Y Allison
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

George W Webb, III
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jacob Gantz
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John D. van Loben Sels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jonathan D. Baker
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Justin F. Boyce
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Michael J. Modl
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Robert D. Rhoad
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Steven R. Daniels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Valerie Margo Wagner
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

W. Bryan Farney
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Andrew Clarkowski
(See above for address)
ATTORNEY TO BE NOTICED

V.

Counter Defendant

HEWLETT-PACKARD COMPANY represented by **Brian M. Fogarty**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Brooke A. Beros
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

David L. Alberti
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

David K. Wiggins
(See above for address)
LEAD ATTORNEY

ATTORNEY TO BE NOTICED

Edward H. Sikorski
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

James Donald Peterson
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John Allcock
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Kathryn Bridget Riley
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Mark D. Fowler
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Nikki Wyll
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Richard Thomas Mulloy
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Sal Lim
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Sean C. Cunningham
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Counter Claimant

Acer, Inc. represented by **Connie Merriett**
(See above for address)

LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Craig Y Allison
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

George W Webb, III
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jacob Gantz
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John D. van Loben Sels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jonathan D. Baker
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Justin F. Boyce
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Michael J. Modl
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Robert D. Rhoad
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Steven R. Daniels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Valerie Margo Wagner
(See above for address)

LEAD ATTORNEY
ATTORNEY TO BE NOTICED

W. Bryan Farney
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Andrew Clarkowski
(See above for address)
ATTORNEY TO BE NOTICED

Counter Claimant

Acer America Corporation represented by **Connie Merriett**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Craig Y Allison
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

George W Webb, III
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jacob Gantz
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John D. van Loben Sels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jonathan D. Baker
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Justin F. Boyce
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Michael J. Modl
(See above for address)

LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Robert D. Rhoad
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Steven R. Daniels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Valerie Margo Wagner
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

W. Bryan Farney
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Andrew Clarkowski
(See above for address)
ATTORNEY TO BE NOTICED

V.

Counter Defendant

HEWLETT-PACKARD COMPANY represented by **Brian M. Fogarty**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Brooke A. Beros
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

David L. Alberti
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

David K. Wiggins
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Edward H. Sikorski
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

James Donald Peterson
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John Allcock
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Kathryn Bridget Riley
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Mark D. Fowler
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Nikki Wyll
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Richard Thomas Mulloy
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Sal Lim
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Sean C. Cunningham
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Counter Claimant

HEWLETT-PACKARD COMPANY represented by **Brian M. Fogarty**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Brooke A. Beros
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

David L. Alberti
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

David K. Wiggins
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Edward H. Sikorski
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

James Donald Peterson
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John Allcock
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Kathryn Bridget Riley
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Mark D. Fowler
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Nikki Wyll
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Richard Thomas Mulloy
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Sal Lim
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Sean C. Cunningham
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

V.

Counter Defendant

Acer, Inc. represented by

Connie Merriett
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Craig Y Allison
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

George W Webb, III
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jacob Gantz
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John D. van Loben Sels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jonathan D. Baker
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Justin F. Boyce
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Michael J. Modl

(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Robert D. Rhoad
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Steven R. Daniels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Valerie Margo Wagner
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

W. Bryan Farney
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Andrew Clarkowski
(See above for address)
ATTORNEY TO BE NOTICED

Counter Defendant

Acer America Corporation represented by **Connie Merriett**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Craig Y Allison
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

George W Webb, III
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jacob Gantz
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John D. van Loben Sels

(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jonathan D. Baker
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Justin F. Boyce
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Michael J. Modl
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Robert D. Rhoad
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Steven R. Daniels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Valerie Margo Wagner
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

W. Bryan Farney
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Andrew Clarkowski
(See above for address)
ATTORNEY TO BE NOTICED

Counter Claimant

Acer, Inc. represented by **Connie Merriett**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Craig Y Allison

(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

George W Webb, III
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jacob Gantz
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John D. van Loben Sels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jonathan D. Baker
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Justin F. Boyce
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Michael J. Modl
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Robert D. Rhoad
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Steven R. Daniels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Valerie Margo Wagner
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

W. Bryan Farney

(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Andrew Clarkowski
(See above for address)
ATTORNEY TO BE NOTICED

Counter Claimant

Acer America Corporation represented by **Connie Merriett**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Craig Y Allison
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

George W Webb, III
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jacob Gantz
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John D. van Loben Sels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jonathan D. Baker
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Justin F. Boyce
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Michael J. Modl
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Robert D. Rhoad

(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Steven R. Daniels
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Valerie Margo Wagner
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

W. Bryan Farney
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Andrew Clarkowski
(See above for address)
ATTORNEY TO BE NOTICED

V.

Counter Defendant

HEWLETT-PACKARD COMPANY represented by **Brian M. Fogarty**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Brooke A. Beros
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

David L. Alberti
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

David K. Wiggins
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Edward H. Sikorski
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

James Donald Peterson
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

John Allcock
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Kathryn Bridget Riley
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Mark D. Fowler
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Nikki Wyll
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Richard Thomas Mulloy
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Sal Lim
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Sean C. Cunningham
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Date Filed	#	Docket Text
10/30/2007	1	COMPLAINT against HEWLETT-PACKARD COMPANY (Filing fee $ 350 receipt number 64422.), filed by all plaintiffs. (Attachments: # 1 Exhibit A - Patent 5,101,478# 2 Exhibit B - Patent 6,075,686# 3 Exhibit C - Patent 5,903,765# 4 Exhibit D - Patent 5,870,613# 5 Exhibit E - Patent 5,410,713# 6 JS-44 Civil Cover Sheet) (rep) (Entered: 11/01/2007)

10/30/2007	2	Corporate Disclosure Statement by Plaintiffs Acer, Inc., Acer America Corporation. (rep) (Entered: 11/01/2007)
10/30/2007		Standard attachments for Judge Barbara B. Crabb sent: NORTC, Briefing Guidelines, Corporate Disclosure Statement, BBC Order on Dispositive Motions. (rep) (Entered: 11/01/2007)
11/16/2007	3	MOTION to Admit Bryan Farney, Jonathan D. Baker, Robert D. Rhoad, Valerie M. Wagner, Craig Y. Allison, JOhn D. van Loben Sels, Jacob A. Gantz and George W. Webb III Pro Hac Vice by Plaintiffs Acer, Inc., Acer America Corporation. Motions referred to Magistrate Judge Stephen L Crocker. (elc) (Entered: 11/19/2007)
11/19/2007	4	STIPULATION for Extension of time to respond to complaint by Defendant HEWLETT-PACKARD COMPANY, Plaintiffs Acer, Inc., Acer America Corporation. (elc) (Entered: 11/20/2007)
11/23/2007	5	** TEXT ONLY ORDER ** Order Granting 3 MOTION to Admit Bryan Farney, Jonathan D. Baker, Robert D. Rhoad, Valerie M. Wagner, Craig Y. Allison, JOhn D. van Loben Sels, Jacob A. Gantz and George W. Webb III Pro Hac Vice filed by Acer America Corporation, Acer, Inc. Signed by Magistrate Judge Stephen L Crocker on 11/20/07. (elc) (Entered: 11/23/2007)
11/26/2007	6	** TEXT ONLY ORDER ** Order Granting 4 STIPULATION for Extension of time to respond to complaint. Answer due by 12/10/2007. Signed by Magistrate Judge Stephen L Crocker on 11/26/07. (elc) (Entered: 11/26/2007)
12/10/2007	7	First MOTION to Admit Kathleen B. Riley, Edward H. Sikorski, Sean C. Cunningham, Richard T. Mulloy, David L. Alberti, Brooke A. Beros, and Brian M. Fogerty of DLA Piper US LLP Pro Hac Vice by Defendant HEWLETT-PACKARD COMPANY. Motions referred to Magistrate Judge Stephen L Crocker. (Peterson, James),(ps) (Entered: 12/10/2007)
12/10/2007	8	AFFIDAVIT of Kathleen B. Riley re 7 First MOTION to Admit Kathleen B. Riley, Edward H. Sikorski, Sean C. Cunningham, Richard T. Mulloy, David L. Alberti, Brooke A. Beros, and Brian M. Fogerty of DLA Piper US LLP Pro Hac Vice. (Peterson, James),(ps) (Entered: 12/10/2007)
12/10/2007	9	AFFIDAVIT of Edward H. Sikorski re 7 First MOTION to Admit Kathleen B. Riley, Edward H. Sikorski, Sean C. Cunningham, Richard T. Mulloy, David L. Alberti, Brooke A. Beros, and Brian M. Fogerty of DLA Piper US LLP Pro Hac Vice. (Peterson, James),(ps) (Entered: 12/10/2007)
12/10/2007	10	AFFIDAVIT of Sean C. Cunningham re 7 First MOTION to Admit Kathleen B. Riley, Edward H. Sikorski, Sean C. Cunningham, Richard T. Mulloy, David L. Alberti, Brooke A. Beros, and Brian M. Fogerty of DLA Piper US LLP Pro Hac Vice. (Peterson, James),(ps) (Entered: 12/10/2007)

12/10/2007	11	AFFIDAVIT of Richard T. Mulloy re 7 First MOTION to Admit Kathleen B. Riley, Edward H. Sikorski, Sean C. Cunningham, Richard T. Mulloy, David L. Alberti, Brooke A. Beros, and Brian M. Fogerty of DLA Piper US LLP Pro Hac Vice. (Peterson, James),(ps) (Entered: 12/10/2007)
12/10/2007	12	AFFIDAVIT of David L. Alberti re 7 First MOTION to Admit Kathleen B. Riley, Edward H. Sikorski, Sean C. Cunningham, Richard T. Mulloy, David L. Alberti, Brooke A. Beros, and Brian M. Fogerty of DLA Piper US LLP Pro Hac Vice. (Peterson, James),(ps) (Entered: 12/10/2007)
12/10/2007	13	AFFIDAVIT of Brooke A. Beros re 7 First MOTION to Admit Kathleen B. Riley, Edward H. Sikorski, Sean C. Cunningham, Richard T. Mulloy, David L. Alberti, Brooke A. Beros, and Brian M. Fogerty of DLA Piper US LLP Pro Hac Vice. (Peterson, James),(ps) (Entered: 12/10/2007)
12/10/2007	14	AFFIDAVIT of Brian M. Fogarty re 7 First MOTION to Admit Kathleen B. Riley, Edward H. Sikorski, Sean C. Cunningham, Richard T. Mulloy, David L. Alberti, Brooke A. Beros, and Brian M. Fogerty of DLA Piper US LLP Pro Hac Vice. (Peterson, James),(ps) (Entered: 12/10/2007)
12/10/2007	15	CERTIFICATE OF SERVICE by Defendant HEWLETT-PACKARD COMPANY re 7 First MOTION to Admit Kathleen B. Riley, Edward H. Sikorski, Sean C. Cunningham, Richard T. Mulloy, David L. Alberti, Brooke A. Beros, and Brian M. Fogerty of DLA Piper US LLP Pro Hac Vice (Peterson, James),(ps) (Entered: 12/10/2007)
12/10/2007	16	First MOTION for Leave to File *Documents Under Seal*, MOTION to Seal by Defendant HEWLETT-PACKARD COMPANY. Motions referred to Magistrate Judge Stephen L Crocker. (Peterson, James),(ps) (Entered: 12/10/2007)
12/10/2007	17	First MOTION to Dismiss *Count 1 of Plaintiffs' Complaint, or in the Alternative, For Summary Judgment* by Defendant HEWLETT-PACKARD COMPANY.Brief in Opposition due by 12/31/2007.,Brief in Reply due by 1/10/2008. (Peterson, James),(ps) (Entered: 12/10/2007)
12/10/2007	18	BRIEF in Support by Defendant HEWLETT-PACKARD COMPANY re: 17 First MOTION to Dismiss *Count 1 of Plaintiffs' Complaint, or in the Alternative, For Summary Judgment* filed by HEWLETT-PACKARD COMPANY (Sealed Document) (Peterson, James),(ps) (Entered: 12/10/2007)
12/10/2007	19	AFFIDAVIT of Kathleen Riley re 17 First MOTION to Dismiss *Count 1 of Plaintiffs' Complaint, or in the Alternative, For Summary Judgment* (Sealed Document). (Peterson, James),(ps) (Entered: 12/10/2007)
12/10/2007	20	AFFIDAVIT of Joseph W. Beyers re 17 First MOTION to Dismiss *Count 1 of Plaintiffs' Complaint, or in the Alternative, For Summary Judgment* (Sealed Document). (Peterson, James),(ps) (Entered: 12/10/2007)
12/10/2007	21	NOTICE by Defendant HEWLETT-PACKARD COMPANY re 17 First

		MOTION to Dismiss *Count 1 of Plaintiffs' Complaint, or in the Alternative, For Summary Judgment ; Request for Judicial Notice in Support of Motion* (Peterson, James)Contacted attorney; asked to refile document using correct event. Modified on 12/11/2007 (Jacobson, Kris). (Entered: 12/10/2007)
12/10/2007	22	CERTIFICATE OF SERVICE by Defendant HEWLETT-PACKARD COMPANY re [18] Brief in Support, 16 First MOTION for Leave to File *Documents Under Seal* MOTION to Seal, 21 Notice (Other), 17 First MOTION to Dismiss *Count 1 of Plaintiffs' Complaint, or in the Alternative, For Summary Judgment*, [20] Affidavit, [19] Affidavit (Peterson, James),(ps) (Entered: 12/10/2007)
12/10/2007	23	ANSWER to Complaint with Jury Demand, COUNTERCLAIM against Acer, Inc., Acer America Corporation by Defendant HEWLETT-PACKARD COMPANY, Plaintiffs Acer, Inc., Acer America Corporation. (Attachments: # 1 Exhibit Exhibit A# 2 Exhibit Exhibit B# 3 Exhibit Exhibit C# 4 Exhibit Exhibit D) (Peterson, James),(ps) (Entered: 12/10/2007)
12/10/2007	24	CERTIFICATE OF SERVICE by Defendant HEWLETT-PACKARD COMPANY re 23 Answer to Complaint,, Counterclaim, (Peterson, James),(ps) (Entered: 12/10/2007)
12/11/2007	25	MOTION Request for Judicial Notice re 17 First MOTION to Dismiss *Count 1 of Plaintiffs' Complaint, or in the Alternative, For Summary Judgment* by Defendant HEWLETT-PACKARD COMPANY. Motions referred to Magistrate Judge Stephen L Crocker. (Peterson, James) (Entered: 12/11/2007)
12/11/2007	26	** TEXT ONLY ORDER ** Order Granting 16 First MOTION for Leave to File *Documents Under Seal* MOTION to Seal filed by HEWLETT-PACKARD COMPANY. Signed by Magistrate Judge Stephen L Crocker on 12/11/07. (elc),(ps) (Entered: 12/11/2007)
12/11/2007	27	First MOTION to Admit Mark D. Fowler and John Allcock Pro Hac Vice by Defendant HEWLETT-PACKARD COMPANY, Counter Claimant HEWLETT-PACKARD COMPANY. Motions referred to Magistrate Judge Stephen L Crocker. (Peterson, James) (Entered: 12/11/2007)
12/11/2007	28	AFFIDAVIT of Mark D. Fowler re 27 First MOTION to Admit Mark D. Fowler and John Allcock Pro Hac Vice. (Peterson, James) (Entered: 12/11/2007)
12/11/2007	29	AFFIDAVIT of John Allcock re 27 First MOTION to Admit Mark D. Fowler and John Allcock Pro Hac Vice. (Peterson, James) (Entered: 12/11/2007)
12/14/2007	30	** TEXT ONLY ORDER ** Order Granting 27 First MOTION to Admit Mark D. Fowler and John Allcock Pro Hac Vice filed by HEWLETT-PACKARD COMPANY. Mark D. Fowler for HEWLETT-PACKARD COMPANY, John Allcock

		for HEWLETT-PACKARD COMPANY,HEWLETT-PACKARD COMPANY admitted pro hac vice. Signed by Magistrate Judge Stephen L Crocker on 12/12/07. (krj) (Entered: 12/14/2007)
12/14/2007	31	** TEXT ONLY ORDER ** Order Granting 7 First MOTION to Admit Kathleen B. Riley, Edward H. Sikorski, Sean C. Cunningham, Richard T. Mulloy, David L. Alberti, Brooke A. Beros, and Brian M. Fogerty of DLA Piper US LLP Pro Hac Vice filed by HEWLETT-PACKARD COMPANY. Kathryn B. Riley for HEWLETT-PACKARD COMPANY, Edward H. Sikorski for HEWLETT-PACKARD COMPANY,HEWLETT-PACKARD COMPANY, Sean C. Cunningham for HEWLETT-PACKARD COMPANY,HEWLETT-PACKARD COMPANY, Richard T. Mulloy for HEWLETT-PACKARD COMPANY,HEWLETT-PACKARD COMPANY, David L. Alberti for HEWLETT-PACKARD COMPANY,HEWLETT-PACKARD COMPANY, Brooke A. Beros for HEWLETT-PACKARD COMPANY,HEWLETT-PACKARD COMPANY, Brian M. Fogerty for HEWLETT-PACKARD COMPANY,HEWLETT-PACKARD COMPANY admitted pro hac vice. Signed by Magistrate Judge Stephen L Crocker on 12/12/07. (krj) (Entered: 12/14/2007)
12/17/2007	32	Corporate Disclosure Statement by Defendant HEWLETT-PACKARD COMPANY, Counter Claimant HEWLETT-PACKARD COMPANY. (Peterson, James) (Entered: 12/17/2007)
12/21/2007	33	STIPULATION for Extension of Time to Respond by Counter Defendants Acer, Inc., Acer America Corporation, Defendant HEWLETT-PACKARD COMPANY, Plaintiffs Acer, Inc., Acer America Corporation, Counter Claimant HEWLETT-PACKARD COMPANY. (Modl, Michael) (Entered: 12/21/2007)
12/26/2007	34	** TEXT ONLY ORDER ** Order Granting 33 STIPULATION for Extension of Time to Respond by Counter Defendants Acer, Inc., Acer America Corporation, Defendant HEWLETT-PACKARD COMPANY, Plaintiffs Acer, Inc., Acer America Corporation, Counter Claimant HEWLETT-PACKARD COMPANY filed by HEWLETT-PACKARD COMPANY, Acer America Corporation, Acer, Inc. Response to motion to dismiss due by 1/10/2008; Response to Counterclaim due 1/21/2008. Signed by Magistrate Judge Stephen L Crocker on 12/26/07. (rep) Modified on 12/31/2007 to add response to counterclaim due date. (Jacobson, Kris). (Entered: 12/31/2007)
01/08/2008	35	Joint REPORT of Rule 26(f) Planning Meeting. (Baker, Jonathan) (Entered: 01/08/2008)
01/08/2008	36	CERTIFICATE OF SERVICE by Counter Defendants Acer, Inc., Acer America Corporation, Plaintiffs Acer, Inc., Acer America Corporation re 35 Report of Rule 26(f) Planning Meeting (Baker, Jonathan) (Entered: 01/08/2008)

01/10/2008	37	MOTION for Leave to File *Document Under Seal* by Plaintiffs Acer, Inc., Acer America Corporation. Motions referred to Magistrate Judge Stephen L Crocker. (Wagner, Valerie) (Entered: 01/10/2008)
01/10/2008	38	BRIEF in Opposition by Plaintiffs Acer, Inc., Acer America Corporation re: 37 MOTION for Leave to File *Document Under Seal* filed by Acer America Corporation, Acer, Inc., 17 First MOTION to Dismiss *Count 1 of Plaintiffs' Complaint, or in the Alternative, For Summary Judgment* filed by HEWLETT-PACKARD COMPANY (Sealed Document) (Attachments: # 1 Certificate of Service) (Wagner, Valerie) (Entered: 01/10/2008)
01/10/2008	39	AFFIDAVIT of Valerie M. Wagner filed by Plaintiffs Acer, Inc., Acer America Corporation re: 17 MOTION to Dismiss Count 1 of Plaintiffs' Complaint, or in the Alternative, For Summary Judgment filed by Hewlett-Packard Company. (Attachments: # 1 Exhibit A) (Wagner, Valerie) Modified on 1/11/2008 to add correct docket entry relationship. (Jacobson, Kris). (Entered: 01/10/2008)
01/10/2008	40	Motion for Judicial Notice by Plaintiffs Acer, Inc., Acer America Corporation (Wagner, Valerie) Modified on 1/11/2008 to add correct docket entry relationship. (Jacobson, Kris). Modified on 1/14/2008 to correct event type. (Jacobson, Kris). (Entered: 01/10/2008)
01/10/2008	41	Proposed Findings of Fact by Plaintiffs Acer, Inc., Acer America Corporation re: 17 MOTION to Dismiss Count 1 of Plaintiffs' Complaint, or in the Alternative, For Summary Judgment filed by Hewlett-Packard Company. (Wagner, Valerie) Modified on 1/11/2008 to add correct docket entry relationship. (Jacobson, Kris). (Entered: 01/10/2008)
01/11/2008	42	***DOCUMENT RELATES TO DOCKET #38*** *APPENDIX TO OPPOSITION TO MOTION TO DISMISS* filed by Acer America Corporation, Acer, Inc. (Attachments: # 1 Exhibit 1# 2 Exhibit 2# 3 Exhibit 3# 4 Exhibit 4# 5 Exhibit 5# 6 Exhibit 6# 7 Exhibit 7# 8 Exhibit 8) (Wagner, Valerie) Modified on 1/14/2008 (Jacobson, Kris). (Entered: 01/11/2008)
01/11/2008	43	BRIEF in Opposition by Plaintiffs Acer, Inc., Acer America Corporation re: 25 Motion for Judicial Notice. (Wagner, Valerie) Modified on 1/14/2008 to add correct docket entry relationship. (Jacobson, Kris). (Entered: 01/11/2008)
01/11/2008	44	MOTION - REQUEST FOR JUDICIAL NOTICE filed by Acer America Corporation, Acer, Inc. (Attachments: # 1 Exhibit A) (Wagner, Valerie) Modified on 1/14/2008 to add correct event type. (Jacobson, Kris). (Entered: 01/11/2008)
01/17/2008	45	PRETRIAL CONFERENCE ORDER -Amendments to Pleadings due by 3/14/2008. Dispositive Motions due by 6/27/2008. Motions in Limine due by 9/29/2008. Response to Motion due by 10/10/2008. Final Pretrial Conference set for 10/16/2008 04:00 at PM.Final Pretrial Submissions due 10/9/2008. Jury Selection and Trial set for 10/27/2008 at 09:00 AM. Signed by Magistrate Judge Stephen L Crocker on 1/16/08. (elc)

		(Entered: 01/17/2008)
01/17/2008		Scheduling set re: Claims Construction. Claims Construction Hearing set for 4/18/2008 09:00 at AM. Claims Construction Initial Brief due by 3/28/2008. Claims Construction Response Brief due by 4/10/2008. Signed by Magistrate Judge Stephen L Crocker on 1/16/08. (elc) (Entered: 01/17/2008)
01/18/2008	46	*Acer Inc. and Acer America Corporation's* ANSWER to Counterclaim, First COUNTERCLAIM against HEWLETT-PACKARD COMPANY by Counter Defendants Acer, Inc., Acer America Corporation, Plaintiffs Acer, Inc., Acer America Corporation, Defendant HEWLETT-PACKARD COMPANY, Counter Claimant HEWLETT-PACKARD COMPANY. (Attachments: # 1 Certificate of Service) (Wagner, Valerie) (Entered: 01/18/2008)
01/22/2008	47	Second MOTION to Seal by Defendant HEWLETT-PACKARD COMPANY. Motions referred to Magistrate Judge Stephen L Crocker. (Peterson, James) (Entered: 01/22/2008)
01/22/2008	48	BRIEF in Reply in Support re: 47 Second MOTION to Seal filed by HEWLETT-PACKARD COMPANY, 17 First MOTION to Dismiss *Count 1 of Plaintiffs' Complaint, or in the Alternative, For Summary Judgment* filed by HEWLETT-PACKARD COMPANY (Sealed Document) (Peterson, James) (Entered: 01/22/2008)
01/22/2008	49	Response to Proposed Findings of Fact by Defendant HEWLETT-PACKARD COMPANY re: 17 First MOTION to Dismiss *Count 1 of Plaintiffs' Complaint, or in the Alternative, For Summary Judgment* filed by HEWLETT-PACKARD COMPANY, 47 Second MOTION to Seal filed by HEWLETT-PACKARD COMPANY (Sealed Document) (Peterson, James) (Entered: 01/22/2008)
01/25/2008	50	MOTION for Leave to File *to File Sur-Reply in Opposition to Hewlett-Packard's Motion to Dismiss Count 1, or in the Alternative for Summary Judgment* by Plaintiffs Acer, Inc., Acer America Corporation. Motions referred to Magistrate Judge Stephen L Crocker. (Attachments: # 1 Certificate of Service) (Wagner, Valerie) (Entered: 01/25/2008)
01/25/2008	51	BRIEF in Sur-Reply by Plaintiffs Acer, Inc., Acer America Corporation re: 50 MOTION for Leave to File *to File Sur-Reply in Opposition to Hewlett-Packard's Motion to Dismiss Count 1, or in the Alternative for Summary Judgment* filed by Acer America Corporation, Acer, Inc., 17 First MOTION to Dismiss *Count 1 of Plaintiffs' Complaint, or in the Alternative, For Summary Judgment* filed by HEWLETT-PACKARD COMPANY (Attachments: # 1 Attachment A# 2 Attachment B# 3 Certificate of Service) (Wagner, Valerie) (Entered: 01/25/2008)
01/28/2008	52	** TEXT ONLY ORDER ** Order Granting 37 MOTION for Leave to File *Document Under Seal* filed by Acer America Corporation, Acer, Inc. Signed by Magistrate Judge Stephen L Crocker on 1/28/08. (krj) (Entered: 01/28/2008)

01/28/2008	53	** TEXT ONLY ORDER ** Order Granting 47 Second MOTION to Seal filed by HEWLETT-PACKARD COMPANY. Signed by Magistrate Judge Stephen L Crocker on 1/28/08. (krj) (Entered: 01/28/2008)
02/07/2008	54	ANSWER to Counterclaim by Counter Claimant HEWLETT-PACKARD COMPANY. (Riley, Kathryn) (Entered: 02/07/2008)
02/19/2008	55	STIPULATION for Protective Order by Counter Defendants Acer, Inc., Acer America Corporation, HEWLETT-PACKARD COMPANY, Defendant HEWLETT-PACKARD COMPANY, Counter Claimants Acer, Inc., Acer America Corporation, HEWLETT-PACKARD COMPANY, Plaintiffs Acer, Inc., Acer America Corporation. (Modl, Michael) (Entered: 02/19/2008)
02/19/2008	56	Set Deadlines/Hearings: Attorney Modl responsible for setting up the call.Scheduling Conference set for 2/20/2008 at 11:00 AM. (krj) (Entered: 02/19/2008)
02/19/2008	57	PROTECTIVE ORDER Signed by Magistrate Judge Stephen L Crocker on 2/19/08. (krj) (Entered: 02/19/2008)
02/20/2008		Minute Entry for proceedings held before Judge Stephen L Crocker : Telephone Scheduling Conference held on 2/20/2008 [:15] (cak) (Entered: 02/20/2008)
02/20/2008	58	AMENDED SCHEDULING ORDER:Claims Construction Initial Brief due by 4/25/2008, Claims Construction Response Brief due by 5/8/2008, Dispositive Motions due by 7/25/2008; Claims Construction hearing rescheduled for 5/16/08. Signed by Magistrate Judge Stephen L Crocker on 2/20/08. (krj) (Entered: 02/20/2008)
02/20/2008		Set/Reset Hearings: Claims Construction Hearing reset for 5/16/2008 09:00 at AM. (krj) (Entered: 02/20/2008)
02/22/2008	59	MOTION to Admit Nikki Wyll Pro Hac Vice by Defendant HEWLETT-PACKARD COMPANY. Motions referred to Magistrate Judge Stephen L Crocker. (Peterson, James) (Entered: 02/22/2008)
02/25/2008	60	** TEXT ONLY ORDER ** Order Granting 59 MOTION to Admit Nikki Wyll Pro Hac Vice filed by HEWLETT-PACKARD COMPANY. Nikki Wyll for HEWLETT-PACKARD COMPANY admitted pro hac vice. Signed by Magistrate Judge Stephen L Crocker on 2/25/08. (krj) (Entered: 02/25/2008)
02/27/2008	61	MOTION to Admit Sal Lim Pro Hac Vice by Defendant HEWLETT-PACKARD COMPANY. Motions referred to Magistrate Judge Stephen L Crocker. (Peterson, James) (Entered: 02/27/2008)
02/27/2008	62	** TEXT ONLY ORDER ** Order Granting 61 MOTION to Admit Sal Lim Pro Hac Vice filed by HEWLETT-PACKARD COMPANY. Signed by Magistrate Judge Stephen L Crocker on 2/27/08. (krj) (Entered: 02/27/2008)

03/21/2008	63	MOTION to Compel *Hewlett-Packard to produce Certain Categories of High Priority Documents by April 4, 2008* by Plaintiffs Acer, Inc., Acer America Corporation. Motions referred to Magistrate Judge Stephen L Crocker.Response due by 3/28/2008. (Wagner, Valerie) (Entered: 03/21/2008)
03/21/2008	64	**DISREGARD - INCORRECT EVENT TYPE, RE-FILED AS DKT. #68** MOTION to Compel *Memorandum of Points and Authorities in Support of Motion to Compel*, filed by Plaintiffs Acer America Corporation, Acer, Inc. Motion referred to Magistrate Judge Stephen L Crocker. Response due by 3/28/2008. (Wagner, Valerie) Modified on 3/24/2008 (Wiseman, Andrew). (Entered: 03/21/2008)
03/21/2008	65	AFFIDAVIT of Valerie M. Wagner re: 63 MOTION to Compel *Hewlett-Packard to produce Certain Categories of High Priority Documents*, filed by Acer America Corporation, Acer, Inc. (Attachments: # 1 Exhibit A # 2 Exhibit B # 3 Exhibit C # 4 Exhibit D # 5 Exhibit E # 6 Exhibit F # 7 Exhibit G # 8 Exhibit H # 9 Exhibit I # 10 Exhibit J # 11 Exhibit K # 12 Exhibit L) (Wagner, Valerie) Modified docket text on 3/24/2008 (Wiseman, Andrew). (Entered: 03/21/2008)
03/21/2008	66	CERTIFICATE OF SERVICE by Plaintiffs Acer America Corporation, Acer, Inc. re 63 MOTION to Compel *Hewlett-Packard to produce Certain Categories of High Priority Documents*, 68 Brief in Support of MOTION to Compel (Wagner, Valerie) Modified docket text on 3/24/2008 (Wiseman, Andrew). (Entered: 03/21/2008)
03/24/2008	67	Set/Reset Deadlines as to 63 MOTION to Compel. Motion Hearing set for 3/31/2008 at 1:30 PM. Counsel for Plaintiff responsible for setting up the call. (arw) Modified docket text on 3/24/2008 (Wiseman, Andrew). (Entered: 03/24/2008)
03/24/2008	68	BRIEF in Support by Plaintiffs Acer, Inc., Acer America Corporation re: 63 MOTION to Compel *Hewlett-Packard to produce Certain Categories of High Priority Documents by April 4, 2008* filed by Acer America Corporation, Acer, Inc. (Wagner, Valerie) (Entered: 03/24/2008)
03/25/2008	69	MOTION to Admit David K. Wiggins Pro Hac Vice by Defendant HEWLETT-PACKARD COMPANY. Motions referred to Magistrate Judge Stephen L Crocker. (Attachments: # 1 Exhibit Declaration of David K. Wiggins) (Peterson, James) (Entered: 03/25/2008)
03/25/2008	70	** TEXT ONLY ORDER ** Granting 69 MOTION to Admit David K. Wiggins pro hac vice, filed by HEWLETT-PACKARD COMPANY. David K. Wiggins for HEWLETT-PACKARD COMPANY admitted pro hac vice. Signed by Magistrate Judge Stephen L Crocker on 3/25/2008. (arw) (Entered: 03/25/2008)
03/28/2008	71	BRIEF in Opposition by Defendant HEWLETT-PACKARD COMPANY re: 63 MOTION to Compel *Hewlett-Packard to produce Certain Categories of High Priority Documents by April 4, 2008* filed by Acer

		America Corporation, Acer, Inc. (Riley, Kathryn) (Entered: 03/28/2008)
03/28/2008	72	AFFIDAVIT of Kathryn B. Riley filed by Defendant HEWLETT-PACKARD COMPANY *in Opposition to Acer's* re: 63 MOTION to Compel *Hewlett-Packard to produce Certain Categories of High Priority Documents by April 4, 2008* filed by Acer America Corporation, Acer, Inc. (Attachments: # 1 Exhibit 1# 2 Exhibit 2# 3 Exhibit 3# 4 Exhibit 4# 5 Exhibit 5# 6 Exhibit 6# 7 Exhibit 7# 8 Exhibit 8# 9 Exhibit 9# 10 Exhibit 10# 11 Exhibit 11# 12 Exhibit 12# 13 Exhibit 13# 14 Exhibit 14# 15 Exhibit 15# 16 Exhibit 16# 17 Exhibit 17# 18 Exhibit 18) (Riley, Kathryn) (Entered: 03/28/2008)
03/28/2008	73	CERTIFICATE OF SERVICE by Defendant HEWLETT-PACKARD COMPANY re 71 Brief in Opposition (Riley, Kathryn) (Entered: 03/28/2008)
03/31/2008	74	ORDER denying as moot 63 MOTION to Compel, signed by Magistrate Judge Stephen L Crocker on 3/31/2008. (arw) (Entered: 04/01/2008)
04/10/2008	75	ORDER granting 17 Motion for Summary Judgment dismissingplaintiffs claim for infringement of United States Patent No. 5,101,478. Signed by Judge Barbara B Crabb on 4/9/08. (elc) (Entered: 04/10/2008)
04/11/2008	76	First AMENDED ANSWER to 1 Complaint,, Amended COUNTERCLAIM against all plaintiffs by Defendant HEWLETT-PACKARD COMPANY, Plaintiffs Acer, Inc., Acer America Corporation. (Attachments: # 1 Exhibit A# 2 Exhibit B# 3 Exhibit C# 4 Exhibit D) (Riley, Kathryn) (Entered: 04/11/2008)
04/11/2008	77	AMENDED COMPLAINT *FOR PATENT INFRINGEMENT, PERMANENT INJUNCTION AND DAMAGES; DEMAND FOR JURY TRIAL* against HEWLETT-PACKARD COMPANY, filed by Acer, Inc., Acer America Corporation. (Attachments: # 1 Exhibit B# 2 Exhibit C# 3 Exhibit D# 4 Exhibit E# 5 Certificate of Service) (Baker, Jonathan) (Entered: 04/11/2008)
04/11/2008	78	ANSWER to Complaint with Jury Demand, COUNTERCLAIM *ACER'S AMENDED REPLY AND COUNTERCLAIMS TO HEWLETT-PACKARD'S COUNTERCLAIMS* against HEWLETT-PACKARD COMPANY by Counter Defendants Acer, Inc., Acer America Corporation, Plaintiffs Acer, Inc., Acer America Corporation, Defendant HEWLETT-PACKARD COMPANY, Counter Claimant HEWLETT-PACKARD COMPANY. (Attachments: # 1 Certificate of Service) (Baker, Jonathan) (Entered: 04/11/2008)
04/15/2008	79	ORDER issuing Letter Rogatory requesting Judicial Assistance to obtain documentary evidence from Lit-On Technology Corporation. Signed by Magistrate Judge Stephen L Crocker on 4/14/08. (elc) (Entered: 04/15/2008)
04/15/2008	80	ORDER issuing Letter Rogatory requesting Judicial Assistance to obtain testimony from Lite-On Technology Corporation. Signed by Magistrate Judge Stephen L Crocker on 4/14/08. (elc) (Entered: 04/15/2008)

04/15/2008	81	ORDER issuing Letter Rogatory requesting Judicial Assistance to obtain Testimony from Bestec Power Electronics Co., LTD. Signed by Magistrate Judge Stephen L Crocker on 4/14/08. (elc) (Entered: 04/15/2008)
04/15/2008	82	ORDER issuing Letter Rogatory requesting Judicial Assistance to obtain documentary evidence from Bestec Power Electronics Cp., LTD. Signed by Magistrate Judge Stephen L Crocker on 4/14/08. (elc) (Entered: 04/15/2008)
04/15/2008	83	ORDER issuing Letter Rogatory requesting Judicial Assistance to Obtain Documentary Evidence from Delta Electronics, Inc. Signed by Magistrate Judge Stephen L Crocker on 4/14/08. (elc) (Entered: 04/15/2008)
04/15/2008	84	ORDER issuing Letter Rogatory requesting Judicial Assistance to Obtain Testimony From Delta Electronics, Inc. Signed by Magistrate Judge Stephen L Crocker on 4/14/08. (elc) (Entered: 04/15/2008)
04/15/2008	85	ORDER issuing Letter Rogatory Requesting Judicial Assistance to Obtain Testimony From RealTek Semiconductor Corporation. Signed by Magistrate Judge Stephen L Crocker on 4/14/08. (elc) (Entered: 04/15/2008)
04/15/2008	86	ORDER issuing Letter Rogatory Requesting Judicial Assistance to Obtain Documentary Evidence from RealTek Semiconductor Corporation. Signed by Magistrate Judge Stephen L Crocker on 4/14/08. (elc) (Entered: 04/15/2008)
04/24/2008	87	ANSWER to Counterclaim, COUNTERCLAIM *ACER'S REPLY AND COUNTERCLAIMS TO HEWLETT-PACKARD'S FIRST AMENDED COUNTERCLAIMS* against HEWLETT-PACKARD COMPANY by Counter Defendants Acer, Inc., Acer America Corporation, Plaintiffs Acer, Inc., Acer America Corporation, Defendant HEWLETT-PACKARD COMPANY, Counter Claimant HEWLETT-PACKARD COMPANY. (Attachments: # 1 Certificate of Service) (Baker, Jonathan) (Entered: 04/24/2008)
04/24/2008	88	**DISREGARD; REPLACED BY DOCUMENT #89** STIPULATION for Waiver of Rights on Appeal and Waiver of Rights to Seek Attorneys' Fees or Sanctions Regarding U.S. Patent No. 5,101,478 by Plaintiffs Acer, Inc., Acer America Corporation. (Wagner, Valerie) ATTORNEY CONTACTED; ASKED TO REFILE USING MATCHING LOG-IN AND SIGNATURE. Modified on 4/25/2008 (Jacobson, Kris). Modified on 5/14/2008 to term. (Kamke, Lynn). (Entered: 04/24/2008)
04/25/2008	89	STIPULATION for Waiver of Rights on Appeal, Waiver to Seek Attorneys' Fees or Sanctions by Counter Defendants Acer, Inc., Acer America Corporation, HEWLETT-PACKARD COMPANY, Acer, Inc., Acer America Corporation, HEWLETT-PACKARD COMPANY, HEWLETT-PACKARD COMPANY, Defendant HEWLETT-PACKARD COMPANY, Counter Claimants Acer, Inc., Acer America

		Corporation, HEWLETT-PACKARD COMPANY, Acer, Inc., Acer America Corporation, Acer, Inc., Acer America Corporation, HEWLETT-PACKARD COMPANY, Plaintiffs Acer, Inc., Acer America Corporation. (Modl, Michael) (Entered: 04/25/2008)
04/25/2008	90	** TEXT ONLY ORDER ** Order approving 89 STIPULATION for Waiver of Rights on Appeal, Waiver to Seek Attorneys' Fees or Sanctions by Counter Defendants Acer, Inc., Acer America Corporation, HEWLETT-PACKARD COMPANY, Acer, Inc., Acer America Corporation, HEWLETT-PACKARD COMPANY, HEWLETT-PACK filed by HEWLETT-PACKARD COMPANY, Acer America Corporation, Acer, Inc. Signed by Judge Barbara B Crabb on 4/25/08. (krj) (Entered: 04/25/2008)
04/25/2008	91	File History of Patent 5,861,864. (krj) (Entered: 04/25/2008)
04/25/2008	92	File History of Patent 6,757,002. (krj) (Entered: 04/25/2008)
04/25/2008	93	File History of Patent 6,678,830 (Volume 1 of 3). (krj) (Entered: 04/25/2008)
04/25/2008	94	File History of Patent 6,678,830 (Volume 2 of 3). (krj) (Entered: 04/25/2008)
04/25/2008	95	File History of Patent 6,678,830 (Volume 3 of 3). (krj) (Entered: 04/25/2008)
04/25/2008	96	*Defendant Hewlett-Packard Company's* ANSWER to Amended Complaint *and*, COUNTERCLAIM *S* against Acer, Inc., Acer America Corporation by Defendant HEWLETT-PACKARD COMPANY, Plaintiffs Acer, Inc., Acer America Corporation. (Mulloy, Richard) (Entered: 04/25/2008)
04/25/2008	97	*Defendant Hewlett-Packard Company's Reply to Acer's Amended Counterclaims* ANSWER to Counterclaim by Defendant HEWLETT-PACKARD COMPANY. (Mulloy, Richard) (Entered: 04/25/2008)
04/25/2008	98	**DISREGARD; REPLACED BY DOCUMENT #106** Claims Construction Submission by Defendant HEWLETT-PACKARD COMPANY, Counter Claimant HEWLETT-PACKARD COMPANY. (Mulloy, Richard) ATTORNEY CONTACTED; ASKED TO REFILE USING CORRECT EVENT. Modified on 4/28/2008 (Jacobson, Kris). Modified on 4/29/2008 (Jacobson, Kris). (Entered: 04/25/2008)
04/25/2008	99	AFFIDAVIT of Richard T. Mulloy re 98 Claims Construction Submission *Declaration of Richard T. Mulloy in Support of Defendant and Counterclaimant Hewlett-Packard Company's Opening Claim Construction Brief*. (Attachments: # 1 Exhibit 1, US Patent No. 5,861,864, pp. 29# 2 Exhibit 2, US Patent No. 6,757,002, pp. 15# 3 Exhibit 3, US Patent No. 6,678,830, pp. 18# 4 Exhibit 4, US Patent No. 5,410,713, pp. 11# 5 Exhibit 5, US Patent No. 5,870,613, pp. 12# 6 Exhibit 6, US Patent No. 5,903,765, pp.12# 7 Exhibit 7, US Patent No. 5,815,141, pp.13) (Mulloy, Richard) Modified docket text on 4/28/2008

		(Jacobson, Kris). (Entered: 04/25/2008)
04/25/2008	100	AFFIDAVIT of Jack D. Grimes re 98 Claims Construction Submission *Declaration of Jack D. Grimes, PH.D., in Support of Hewlett-Packard Company's Opening Claim Construction Brief.* (Attachments: # 1 Exhibit 1, Grimes Resume, pp.19) (Mulloy, Richard) (Entered: 04/25/2008)
04/25/2008	101	AFFIDAVIT of Robert L. Stevenson re 98 Claims Construction Submission *Declaration of Robert L. Stevenson in Support of Hewlett-Packard Company's Opening Claim Construction Brief.* (Attachments: # 1 Exhibit 1, Curriculum Vitae, pp. 22# 2 Exhibit 2, Computer Dictionary, pp. 7# 3 Exhibit 3, IEEE Electronics Terms, pp. 3# 4 Exhibit 4, MCS 51 Microcontroller User Manual, pp. 3) (Mulloy, Richard) (Entered: 04/25/2008)
04/25/2008	102	AFFIDAVIT of Nathaniel Polish re 98 Claims Construction Submission *Declaration of Nathaniel Polish, PH.D. in Support of Hewlett-Packard Company's Opening Claim Construction Brief.* (Attachments: # 1 Exhibit 1, Polish Resume, pp. 8# 2 Exhibit 2, IBM Dictionary of Computing, pp. 4# 3 Exhibit 3, Microsoft Computer Dictionary, pp. 4) (Mulloy, Richard) (Entered: 04/25/2008)
04/25/2008	103	CERTIFICATE OF SERVICE by Defendant HEWLETT-PACKARD COMPANY, Counter Claimant HEWLETT-PACKARD COMPANY re 102 Affidavit, 101 Affidavit, 99 Affidavit,, 98 Claims Construction Submission, 100 Affidavit, (Mulloy, Richard) (Entered: 04/25/2008)
04/25/2008	104	Claims Construction Initial Brief by Plaintiffs Acer, Inc., Acer America Corporation (Attachments: # 1 Certificate of Service) (Baker, Jonathan) (Entered: 04/25/2008)
04/25/2008	105	AFFIDAVIT of Jonathan D. Baker filed by Plaintiffs Acer, Inc., Acer America Corporation re: 104 Claims Construction Initial Brief filed by Acer America Corporation, Acer, Inc. (Attachments: # 1 Exhibit 1-5# 2 Exhibit 6# 3 Exhibit 7-10# 4 Exhibit 11-15# 5 Exhibit 16-30# 6 Exhibit 31-40# 7 Exhibit 41# 8 Exhibit 42-47) (Baker, Jonathan) (Entered: 04/25/2008)
04/28/2008	106	**REPLACES DOCUMENT #98** Claims Construction Initial Brief by Defendant HEWLETT-PACKARD COMPANY, Counter Claimant HEWLETT-PACKARD COMPANY (Mulloy, Richard) Modified on 4/29/2008 (Jacobson, Kris). (Entered: 04/28/2008)
05/07/2008	107	File History of Patent 5,410,713. (vob) (Entered: 05/08/2008)
05/07/2008	108	File History of Patent 5,870,613. (vol. 1) (vob) (Entered: 05/08/2008)
05/07/2008	109	File History of Patent 5,870,613. (vol. 2) (vob) (Entered: 05/08/2008)
05/07/2008	110	File History of Patent 5,903,765. (vob) (Entered: 05/08/2008)
05/07/2008	111	File History of Patent 7,075,686. (vob) (Entered: 05/08/2008)

05/08/2008	112	ANSWER to Counterclaim, COUNTERCLAIM *ACER'S REPLY AND COUNTERCLAIMS TO HEWLETT-PACKARD'S COUNTERCLAIMS* against HEWLETT-PACKARD COMPANY by Counter Defendants Acer, Inc., Acer America Corporation, Plaintiffs Acer, Inc., Acer America Corporation, Defendant HEWLETT-PACKARD COMPANY, Counter Claimant HEWLETT-PACKARD COMPANY. (Attachments: # 1 Certificate of Service) (Baker, Jonathan) (Entered: 05/08/2008)
05/08/2008	113	Claims Construction Response Brief by Defendant HEWLETT-PACKARD COMPANY *REPLY CLAIM CONSTRUCTION BRIEF* (Mulloy, Richard) (Entered: 05/08/2008)
05/08/2008	114	AFFIDAVIT of Richard Mulloy for Defendants re 113 Claims Construction Response Brief *DECLARATION OF RICHARD MULLOY IN SUPPORT OF HP'S REPLY CLAIM CONSTRUCTION BRIEF*. (Attachments: # 1 Exhibit 1 to Declaration of Richard Mulloy in support of HP's Reply Claim Construction Brief, Computer Dictionary Excerpt, 3 pages.# 2 Exhibit 2 to Declaration of Richard Mulloy in support of HP's Reply Claim Construction Brief, Webster Dictionary Excerpt, 3 pages.) (Mulloy, Richard) (Entered: 05/08/2008)
05/08/2008	115	AFFIDAVIT of Jack D. Grimes, Ph.D. re 113 Claims Construction Response Brief *SUPPLEMENTAL DECLARATION OF JACK D. GRIMES, PH.D. IN SUPPORT OF HP'S REPLY CLAIM CONSTRUCTION BRIEF*. (Attachments: # 1 Exhibit 1 to Supplemental Declaration of Jack D. Grimes, Ph.D. in support of HP's Reply Claim Construction Brief, 10-17-90 Article on Micro-chip and Processor, 20 pages.# 2 Exhibit 2 to Supplemental Declaration of Jack D. Grimes, Ph.D. in support of HP's Reply Claim Construction Brief, 10-31-90 Article on Processors, 20 pages.) (Mulloy, Richard) (Entered: 05/08/2008)
05/08/2008	116	Claims Construction Response Brief by Plaintiffs Acer, Inc., Acer America Corporation (Attachments: # 1 Certificate of Service) (Baker, Jonathan) (Entered: 05/08/2008)
05/08/2008	117	AFFIDAVIT of Jonathan D. Baker filed by Plaintiffs Acer, Inc., Acer America Corporation re: 116 Claims Construction Response Brief filed by Acer America Corporation, Acer, Inc. (Attachments: # 1 Exhibit 48# 2 Exhibit 49# 3 Exhibit 50# 4 Exhibit 51# 5 Exhibit 52# 6 Exhibit 53# 7 Exhibit 54# 8 Exhibit 55# 9 Exhibit 56# 10 Exhibit 57# 11 Exhibit 58# 12 Exhibit 59# 13 Exhibit 60# 14 Exhibit 61# 15 Exhibit 62# 16 Exhibit 63# 17 Exhibit 64) (Baker, Jonathan) (Entered: 05/08/2008)
05/13/2008	118	**DISREGARD- No CM/ECF Registration attached.** MOTION to Admit Steven R. Daniels, Justin F. Boyce, Connie E. Merriett Pro Hac Vice by Counter Defendants Acer, Inc., Acer America Corporation, Acer, Inc., Acer America Corporation, Acer, Inc., Acer America Corporation, Counter Claimants Acer, Inc., Acer America Corporation, Acer, Inc., Acer America Corporation, Acer, Inc., Acer America Corporation, Acer, Inc., Acer America Corporation, Plaintiffs Acer, Inc., Acer America Corporation. Motions referred to Magistrate

		Judge Stephen L Crocker. (Clarkowski, Andrew) Modified on 5/14/2008 (Balderas, Vivian). Modified on 5/14/2008 (Balderas, Vivian). (Entered: 05/13/2008)
05/14/2008	119	MOTION to Admit Steven R. Daniels, Justin F. Boyce, Connie E. Merriett Pro Hac Vice by Counter Defendants Acer, Inc., Acer America Corporation, Acer, Inc., Acer America Corporation, Acer, Inc., Acer America Corporation, Counter Claimants Acer, Inc., Acer America Corporation, Acer, Inc., Acer America Corporation, Acer, Inc., Acer America Corporation, Acer, Inc., Acer America Corporation, Plaintiffs Acer, Inc., Acer America Corporation. Motions referred to Magistrate Judge Stephen L Crocker. (Attachments: # 1 Exhibit) (Clarkowski, Andrew) (Entered: 05/14/2008)
05/14/2008	120	** TEXT ONLY ORDER ** Order Granting 119 MOTION to Admit Steven R. Daniels, Justin F. Boyce, Connie E. Merriett Pro Hac Vice, filed by Acer America Corporation, Acer, Inc. Signed by Magistrate Judge Stephen L Crocker on 5/14/2008. (lak) (Entered: 05/14/2008)
05/15/2008	121	STIPULATION for Proposed Order of Presentation at Claim Construction Hearing by Plaintiffs Acer, Inc., Acer America Corporation. (Attachments: # 1 Certificate of Service) (Wagner, Valerie) (Entered: 05/15/2008)
05/16/2008	122	Minute Entry for proceedings held before Judge Barbara B Crabb : Claims Construction Hearing held on 5/16/2008 [3:07] (Court Reporter CS.) (krj) (Entered: 05/16/2008)
05/22/2008	123	STIPULATION for Joint Motion to Extend Deadline to Amend Pleadings to Identify Accused Products by Plaintiffs Acer, Inc., Acer America Corporation by Plaintiffs Acer, Inc., Acer America Corporation. (Wagner, Valerie) (Entered: 05/22/2008)
05/22/2008	124	CERTIFICATE OF SERVICE by Plaintiffs Acer, Inc., Acer America Corporation re 123 STIPULATION for Joint Motion to Extend Deadline to Amend Pleadings to Identify Accused Products by Plaintiffs Acer, Inc., Acer America Corporation (Wagner, Valerie) (Entered: 05/22/2008)
05/23/2008	125	** TEXT ONLY ORDER ** Order Granting 123 STIPULATION for Joint Motion to Extend Deadline to Amend Pleadings to Identify Accused Products by Plaintiffs Acer, Inc., Acer America Corporation filed by Acer America Corporation, Acer, Inc. Amendments to Pleadings due by 6/16/2008. Signed by Magistrate Judge Stephen L Crocker on 5/23/08. (mmo) (Entered: 05/23/2008)
06/06/2008	126	STIPULATION of Dismissal *of All Claims and Counterclaims Without Prejudice Pursuant to Settlement Agreement* by Counter Defendants Acer, Inc., Acer America Corporation, HEWLETT-PACKARD COMPANY, Acer, Inc., Acer America Corporation, HEWLETT-PACKARD COMPANY, HEWLETT-PACKARD COMPANY, Acer, Inc., Acer America Corporation, HEWLETT-PACKARD COMPANY,

		Defendant HEWLETT-PACKARD COMPANY, Counter Claimants Acer, Inc., Acer America Corporation, HEWLETT-PACKARD COMPANY, Acer, Inc., Acer America Corporation, Acer, Inc., Acer America Corporation, HEWLETT-PACKARD COMPANY, Acer, Inc., Acer America Corporation, HEWLETT-PACKARD COMPANY, Plaintiffs Acer, Inc., Acer America Corporation. (Mulloy, Richard) (Entered: 06/06/2008)
06/09/2008	127	ORDER dismissing all claims and counterclaims without prejudice pursuant to settlement agreement as set forth in 126 Stipulation of Dismissal, filed by HEWLETT-PACKARD COMPANY, Acer America Corporation, Acer, Inc. Signed by Judge Barbara B Crabb on 6/9/08. (llj) (Entered: 06/09/2008)
06/24/2008	128	TRANSCRIPT of Claims Construction Hearing, held 5/16/08 before Judge Barbara B Crabb. Court Reporter: CS. Please review the court policy regarding electronic transcripts: see Electronic Transcript Instructions and Notice of Intent to Request Redaction. (elc) (Entered: 06/24/2008)

PACER Service Center			
Transaction Receipt			
07/17/2008 17:02:54			
PACER Login:	fb0051	**Client Code:**	23129-tpl
Description:	Docket Report	**Search Criteria:**	3:07-cv-00620-bbc
Billable Pages:	26	**Cost:**	2.08